CHALMERS
UNIVERSITY OF TECHNOLOGY

Implementation of Iterative Learning Control in an electric drive system

Master's thesis in Systems, Control & Mechatronics

Filip Karlsson
Filip Lundberg

Department of Electrical Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2020

Master's thesis 2020

Implementation of Iterative Learning Control in an electric drive system

Filip Karlsson
Filip Lundberg

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Electrical Engineering
Division of systems and control
Chalmers University of Technology
Gothenburg, Sweden 2020

Implementation of Iterative Learning Control in an electric drive system
Filip Karlsson
Filip Lundberg

Supervisor: Jonas Fredriksson, Department of Electrical Engineering
Examiner: Jonas Fredriksson, Department of Electrical Engineering

Master's Thesis 2020
Department of Electrical Engineering
Division of systems and control
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Typeset in LaTeX
Gothenburg, Sweden 2020

Implementation of Iterative Learning Control in an electric drive system
FILIP KARLSSON
FILIP LUNDBERG
Department of Electrical Engineering
Chalmers University of Technology

Abstract

Windshield wiper systems have been an integral part of vehicle safety for a long time. In dual motor wiper systems, synchronization is important to avoid the wipers crashing into each other. One aspect of this is choosing and following motor speed profiles such that required wiper frequency is acquired and crashing is avoided.

In this thesis, we propose a structured way for choosing speed profiles that fulfills the requirements and an automated tuning scheme based on Iterative Learning Control (ILC) for reference tracking. Different ILC algorithms are evaluated in both simulation and in experiments. To interface with the current system, a least-squares splinefit method is used to downsample the updated reference curve.

The proposed method shows good performance in simulations and experiments. However, both performance and computational aspects could be improved if the splinefit method could be avoided. This would however require changing the underlying control system.

Keywords: iterative learning control, ILC, windshield wipers.

Acknowledgements

We would like to express our gratitude to Joel Strand, Jacob Fjellström and Daniel Chädström, our supervisors at Aros Electronics for their excellent support and guidance through the many ins and outs of the wiper system.

We would also like to thank Jonas Fredriksson, our examiner and supervisor at Chalmers University of Technology, for his fantastic guidance throughout this thesis.

Filip Karlsson, Filip Lundberg, Gothenburg, June 2020

Contents

List of Figures

List of Tables

1

Introduction

1.1 Background

Wiper systems have been around for a long time. The first patent for a window-cleaning device was filed by Mary Anderson in 1903 [1]. This device was purely mechanical and was operated by pulling a lever. Today, most vehicles are equipped with a windshield wiper system driven by an electric motor. The windshield wipers give the driver an unobstructed view in varying weather conditions and are therefore essential from a safety perspective. Therefore, there is legislation that regulates minimum requirements on how such systems should operate. For example, in the European Union windshield wiper systems must be able to handle at least 45 sweeps per minute [2].

Traditional wiper systems use a single DC-motor and complex mechanics to control two wipers. This way, the two wipers are always synchronized and cannot clash into each other. An alternative approach is to use two motors, one for each wiper, and control the motion of each wiper individually. A dual motor system reduces the mechanical design complexity, saves space in the vehicle and allows more complex wiper configurations. When each wiper is controlled individually the wiping motion must be synchronized by a controller such that the wipers do not collide.

Such a dual motor wiper system is the main concern of this thesis. The system is used in several different bus models. Because each bus model has different mechanics and mounting configurations, the controller must be tuned differently for each new bus model. Today, this is a manual and time consuming process of adjusting a speed profile for each wiper.

1.1.1 Dual motor wiper system

The dual motor windshield wiper system consists of two wiper arms. Each arm is connected to an electric motor through a gearbox and a linkage. The linkage converts the rotational motion of the motor shaft to a wiping motion of the arm. This allows the motor to rotate in only one direction and still produce a back-and-forth motion of the wiper. A schematic overview of the wiper system is presented in Figure 1.1.

Both motors are controlled by the same microcontroller. The motors are sensorless and the microcontroller estimates the motor speed and position by counting commu-

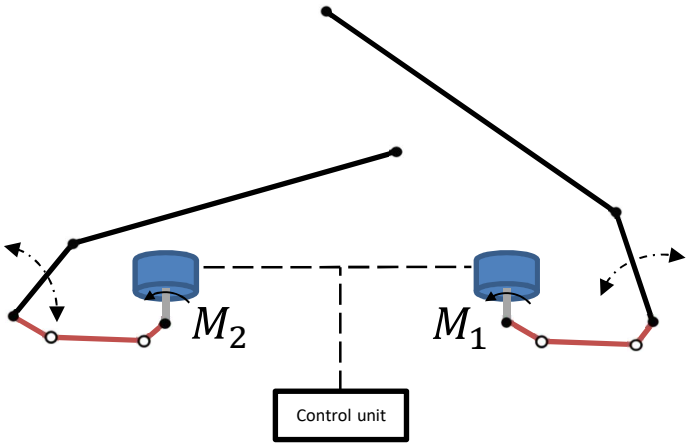


Figure 1.1: Schematic overview of the wiper system.

tator pulses from the motor. There is also a sensor in the gear box that signals when the wiper is in the home position. The speed and position estimates are sometimes erroneous. Therefore, the position estimates are reset at the end of each sweep when the home sensor is signaling.

The speed of each motor is controlled by a PI feedback controller. The reference speed is provided by eight setpoints that are equally spaced over a full wiper sweep. The reference speed to the feedback controller is then given by linear interpolation between these setpoints at the estimated motor angle. The first setpoint specifies the reference speed at sweep angle $\theta = 0$ and the eighth setpoint specifies the reference speed at the sweep angle $\theta = 7\pi/8$. Between $\theta = 7\pi/8$ and $\theta = 2\pi$, the reference speed is interpolated between the eighth and the first setpoint. This is illustrated in Figure 1.2.

The speed profiles are not tracked perfectly by the speed controller. A reason for this is that the motor is experiencing a varying load. The load varies over a wiper sweep with the angle and the speed of the wiper blade. The load also varies over time. For instance, the friction between the windshield and the wiper blade is highly dependent on how wet the windshield is. Wear of the wiper blade and other parts also affects the friction. In addition, the motor resistance varies with temperature. Two buses of the same model may have slightly different properties due to variations in the manufacturing process. Different bus models may have different motors, different mechanics, different wiper blades and different windshields. The mounting configuration also varies between models, in some models the wiper system is mounted upside-down above the windshield. All of these factors affect the dynamics of the system. Thus, a set of speed profiles that work well on a particular bus may therefore not work for a slightly different bus. Therefore, manual tuning is needed for each new bus model.

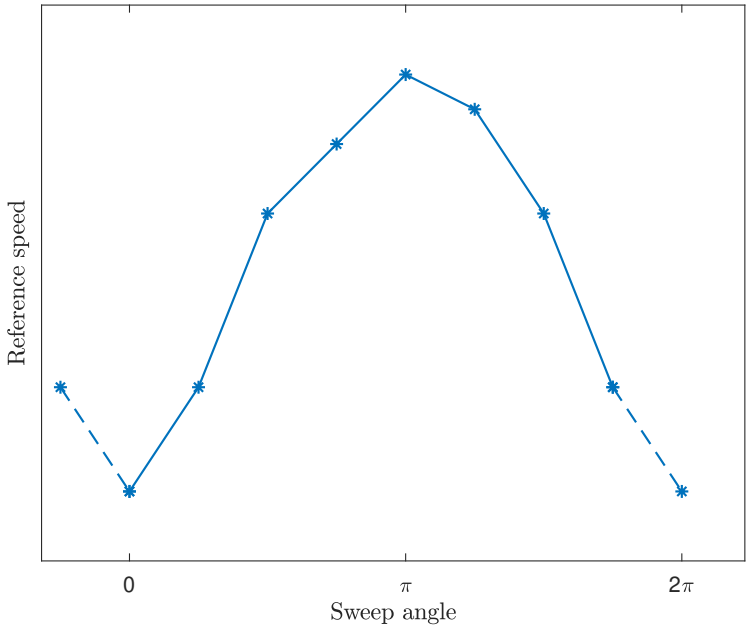


Figure 1.2: Illustration of interpolation between speed setpoints in the wiper control system. The setpoints are marked with stars and the line illustrates the interpolated reference speed.

1.2 Purpose

The purpose of this thesis is to develop a controller that reduces or removes the need for individual tuning for new bus models.

1.3 Aim and scope

The aim of this thesis is to first find velocity profiles that ensure that the wipers do not collide. These trajectory profiles should be usable for multiple bus models. Secondly, the aim is to develop a controller that enables the system to track the velocity profiles independent of some system uncertainties. The idea is to use the cyclic behavior of windshield wiper systems to iteratively improve the tracking of the speed profiles. This can possibly be done with a technique called iterative learning control (ILC). The ILC algorithm should be implemented on the microcontroller available in the current system. The developed algorithm will be tested in simulation and then verified in a test rig.

1.4 Limitations

This thesis work is subject to a number of limitations.

- The current system uses a PI controller to follow velocity profiles. This controller will be left as it is and will not be modified in the project. Instead, the

ILC algorithm will modify the reference to the existing controller.

- In this thesis only ILC is investigated, other methods to improve the tracking performance will not be considered.
- The available test rig has no windshield. The friction caused by the windshield has a large effect on the system in a real application. This will be considered in simulation but not verified in the test rig.

1.5 Thesis outline

In Chapter 2 we start of by modelling the system and its components. Then these models will be used to reason about and calculate a clash-free trajectory for the system.

In Chapter 3 the concept of iterative learning control (ILC) is introduced in a general form and then the investigated algorithms are described.

In Chapter 4 the simulation environment used to investigate the algorithms is described and simulation results are presented.

In Chapter 5 we describe some important considerations when implementing the algorithms on a system with a microcontroller. We then present results from experiments on the test rig and compare the results with the results from the simulations in Chapter 4.

In Chapter 6 we discuss the results found in chapters 4 and 5 and their implications on a system level.

In Chapter 7 final conclusions are made and some future work is suggested.

2

System modeling and trajectory generation

In this chapter we start of by modeling the system and its components. Then this model is used to reason about and calculate a clash-free trajectory for the system.

2.1 Plant model of the wiper system

Below we will define the system in more detail, both for simulation and for control design purposes.

2.1.1 Modeling of linkage mechanism

We start by deriving a kinematic relationship between the gearbox shaft angle θ and the angle of the wiper blade φ. The wiper blade is connected to the gearbox through a four-bar linkage mechanism in a crank and rocker configuration illustrated in Figure 2.1. This allows the motor to rotate in only one direction and still produce a back-and-forth motion of the wiper. It is possible to calculate the rocker angle φ as a function of the crank angle θ by using the kinematic loop closure equation. With ℓ_0, ℓ_1, ℓ_2, ℓ_3 and α defined as in Figure 2.1, the loop closure equation can be written as

$$\ell_0 e^{j0} + \ell_3 e^{j\varphi} = \ell_1 e^{j\theta} + \ell_2 e^{j\alpha}. \tag{2.1}$$

Taking the conjugate of (2.1) gives

$$\ell_0 e^{-j0} + \ell_3 e^{-j\varphi} = \ell_1 e^{-j\theta} + \ell_2 e^{-j\alpha}. \tag{2.2}$$

Multiplying (2.1) and (2.2) and rearranging terms gives

$$\ell_0^2+\ell_1^2+\ell_2^2-\ell_0\ell_1(e^{j\theta}+e^{-j\theta})+\ell_0\ell_3(e^{j\varphi}+e^{-j\varphi})-\ell_1\ell_3(e^{j(\varphi-\theta)}+e^{-j(\varphi-\theta)})-\ell_2^2\underbrace{e^{j(\alpha-\alpha)}}_{=1}=0. \tag{2.3}$$

Then, using $\cos x = \frac{1}{2}(e^{jx} + e^{-jx})$ we get

$$2\ell_0\ell_3\cos\varphi - 2\ell_0\ell_1\cos\theta - 2\ell_1\ell_3\cos(\varphi-\theta) + \ell_0^2 + \ell_1^2 + \ell_3^2 - \ell_2^2 = 0, \tag{2.4}$$

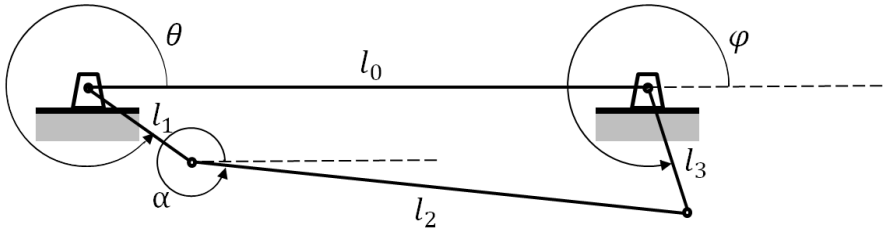


Figure 2.1: Four-bar linkage mechanism.

which can be written as

$$K_1 \cos\varphi - K_2 \cos\theta + K_3 = \cos(\varphi - \theta)\,, \quad \text{where} \begin{cases} K_1 = \ell_0/\ell_1 \\ K_2 = \ell_0/\ell_3 \\ K_3 = \frac{\ell_0^2+\ell_1^2-\ell_2^2+\ell_3^2}{2\ell_1\ell_3}. \end{cases} \tag{2.5}$$

Equation (2.5) is the well-known Freudenstein's equation [3] for four-bar mechanisms. To solve for φ, (2.5) can be rewritten with the trigonometric identity

$$\cos(x - y) = \cos x \cos y + \sin x \sin y$$

giving the form

$$K_1 \cos\varphi - K_2 \cos\theta + K_3 = \cos\varphi \cos\theta + \sin\varphi \sin\theta\,. \tag{2.6}$$

Using the tangent half-angle substitution

$$t = \tan(\frac{\varphi}{2}) \implies \begin{cases} \sin\varphi = \frac{2t}{1+t^2} \\ \cos\varphi = \frac{1-t^2}{1+t^2} \end{cases} \tag{2.7}$$

(2.6) becomes

$$K_1 \frac{1-t^2}{1+t^2} - K_2 \cos\theta + K_3 = \frac{1-t^2}{1+t^2}\cos\theta + \frac{2t}{1+t^2}\sin\theta \tag{2.8}$$

which with some rearranging gives

$$At^2 + Bt + C = 0\,, \quad \text{where} \begin{cases} A = -K_1 + (1 - K_2)\cos\theta + K_3 \\ B = -2\sin\theta \\ C = K_1 - (1 + K_2)\cos\theta + K_3 \end{cases}. \tag{2.9}$$

Equation (2.9) can then be solved with the quadratic formula

$$\varphi(\theta) = 2\arctan\frac{-B \pm \sqrt{B^2 - 4AC}}{2A} \tag{2.10}$$

where a positive sign before the square root gives a so called open configuration on the crank-rocker mechanism, which is what is used in the wiper system.

2.1.2 Modeling of system dynamics

In this section we will model the DC-motor, the gearbox and the wiper blade and then use the kinematic relationship from Section 2.1.1 to create a model that relates the applied motor voltage to the angular velocity of the motor.

The system can be split into two parts, one part that rotates and one part that moves back-and-forth. The rotating part consists of the motor, the gear box and parts of the linkage. The back-and-forth moving part consists of the wiper arm and parts of the linkage.

First, we will model the motor and the gearbox. The system has two identical brushed DC-motors with integrated gearboxes. As such, a standard DC-motor model is used

$$\begin{aligned} L_{\mathrm{m}}\frac{\mathrm{d}i(t)}{\mathrm{d}t} &= u_{\mathrm{A}}(t) - R_{\mathrm{m}}i(t) - \psi\omega_{\mathrm{m}}(t) \\ \tau_{\mathrm{m}} &= \psi i(t)\,, \end{aligned} \tag{2.11}$$

where L_{m} is the motor inductance, R_{m} is the motor resistance, u_{A} is the applied motor voltage, i is the motor current, ψ is the torque constant, ω_{m} is the motor speed and τ_{m} is the motor torque. We then define the speed of the shaft at the output of the gearbox as

$$\omega = \frac{1}{N_{\mathrm{GB}}}\omega_{\mathrm{m}}\,, \tag{2.12}$$

where N_{GB} is the gear ratio.

A torque that corresponds to friction in both gearbox and motor is modeled as being proportional to the gearbox speed

$$\tau_{mf} = -b_{\mathrm{m}}\omega\,. \tag{2.13}$$

The combined mass of the back-and-forth moving part is modeled as a point mass m_φ at a distance r from the rotational center of the wiper arm as shown in Figure 2.2. Gravity acts on the point mass, this corresponds to the force

$$F_{\mathrm{g}} = m_\varphi g, \tag{2.14}$$

where g is the gravitational acceleration constant.

Further, a friction force F_{wf} acts on the wiper blade. The friction force is modeled as being proportional to the speed of the wiper blade with a coefficient b_{w},

$$F_{\mathrm{wf}}(\theta,\omega) = \dot{\varphi}(\theta,\omega)\cdot b_{\mathrm{w}}\,. \tag{2.15}$$

Here, the angular velocity of the wiper blade $\dot{\varphi}$ can be calculated from the velocity of the gearbox shaft ω and the angle of the gearbox shaft θ,

$$\dot{\varphi}(\theta^*,\omega) = \left.\frac{\partial\varphi}{\partial\theta}\right|_{\theta=\theta^*}\omega. \tag{2.16}$$

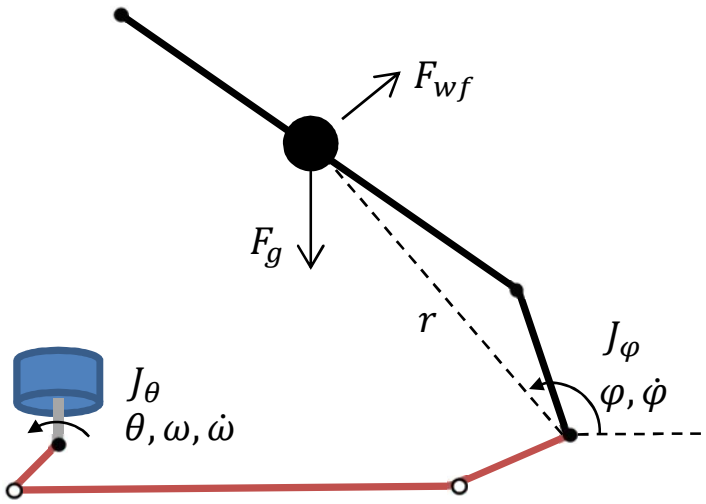


Figure 2.2: One of the wipers with definition.

The forces F_{g} and F_{wf} are transferred through the wiper arm, the linkage and the gearbox and experienced as torques at the motor shaft. The mechanical advantage of the gearbox is constant while the mechanical advantage of the linkage varies with the motor angle. From [3] we know that the ratio between the input and output torque in a linkage or gearbox is given by

$$\frac{\tau_{\mathrm{out}}}{\tau_{\mathrm{in}}} = \frac{\omega_{\mathrm{in}}}{\omega_{\mathrm{out}}}, \tag{2.17}$$

that is, the inverse ratio between the input and output angular velocities. It is then clear that a torque τ_φ applied at the back-and-forth-moving side of the linkage is experienced as an equivalent torque $\tau_{eq,\theta}$ at the motor shaft

$$\tau_{eq,\theta}(\theta^*, \tau_\varphi) = \frac{1}{N_{\mathrm{GB}}} \left.\frac{\partial \varphi}{\partial \theta}\right|_{\theta=\theta^*} \tau_\varphi \,. \tag{2.18}$$

Then, the torque experienced at the motor from the friction force is given by

$$\tau_{\mathrm{wf}}(\theta^*, \omega) = -\frac{1}{N_{\mathrm{GB}}} \left.\frac{\partial \varphi}{\partial \theta}\right|_{\theta=\theta^*} \cdot F_{\mathrm{wf}}(\theta^*, \omega) \cdot r, \tag{2.19}$$

or by inserting (2.15) and (2.16)

$$\tau_{\mathrm{wf}}(\theta^*, \omega) = -\frac{1}{N_{\mathrm{GB}}} \left(\left.\frac{\partial \varphi}{\partial \theta}\right|_{\theta=\theta^*} \right)^2 \cdot r \cdot b_{\mathrm{w}} \cdot \omega. \tag{2.20}$$

Similarly, for the gravity force F_{g}, we get

$$\tau_{\mathrm{g}}(\theta^*) = -\frac{1}{N_{\mathrm{GB}}} \left.\frac{\partial \varphi}{\partial \theta}\right|_{\theta=\theta^*} \cdot F_{\mathrm{g}} \cdot r \cdot \cos\varphi(\theta^*). \tag{2.21}$$

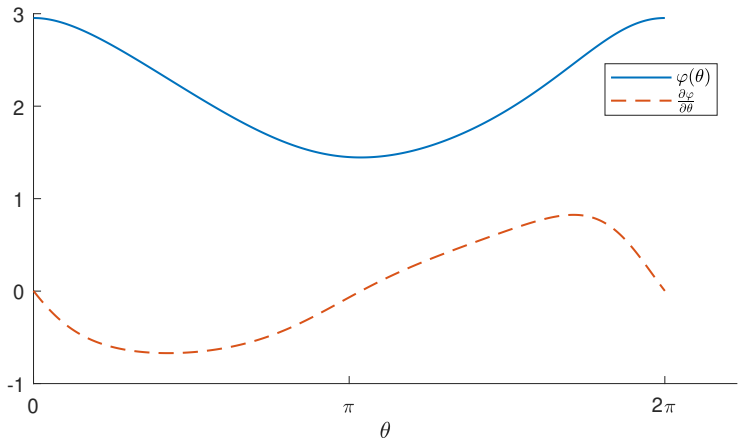


Figure 2.3: The function $\varphi(\theta)$ and $\partial\varphi/\partial\theta$ with the parameters from Table A.1.

Notice that the friction torque always has opposite sign compared to the motor velocity. This means that the friction always has a braking effect on the motor. In contrast, the sign of the gravity torque will change as $\partial\varphi/\partial\theta$ changes sign. By observing the sign of $\partial\varphi/\partial\theta$ in Figure 2.3 and assuming that $\cos\varphi < 0$, one can see that the torque caused by gravity will be working against the motor in the first part of the sweep when the wiper is moving clockwise. When the wiper is moving counter-clockwise on the other hand, the gravity torque will be accelerating the motor and when the wiper changes direction the torque caused by gravity will be zero. Obviously, τ_g will also be zero when $\cos\varphi = 0$, which might happen in some mounting configurations.

Next, in order to complete the model it is necessary to consider the moment of inertia from the two parts of the system. On the rotational side this is trivial, the moment of inertia of the rotating part of the system is constant. We define the constant J_θ which includes the moment of inertia of all rotating parts of the system. The back-and-forth moving part is more interesting. While the moment of inertia J_φ is constant, the moment of inertia experienced by the motor will vary with the motor angle due to the varying mechanical advantage of the linkage. Assuming no losses in the linkage and gearbox, we know from [4] that the reflected moment of inertia at the motor is given by

$$J_{eq,\varphi}(\theta^*) = \left(\frac{1}{N_{\text{GB}}} \left. \frac{\partial\varphi}{\partial\theta} \right|_{\theta=\theta^*} \right)^2 J_\varphi. \tag{2.22}$$

Then, the system's total moment of inertia is given by

$$J_{\text{tot}}(\theta) = J_\theta + J_{eq,\varphi}(\theta). \tag{2.23}$$

Now the complete model becomes

$$L_{\mathrm{m}}\frac{\mathrm{d}i(t)}{\mathrm{d}t} = u_{\mathrm{A}}(t) - R_{\mathrm{m}}i(t) - \psi\omega_{\mathrm{m}}(t) \tag{2.24a}$$

$$J_{\mathrm{tot}}(\theta(t))\frac{\mathrm{d}\omega_{\mathrm{m}}(t)}{\mathrm{d}t} = \psi i(t) + \tau_{\mathrm{mf}}(\omega(t)) + \tau_{\mathrm{wf}}(\theta(t),\omega(t)) + \tau_{\mathrm{g}}(\theta(t)) \tag{2.24b}$$

$$\frac{\mathrm{d}\theta_{\mathrm{m}}(t)}{\mathrm{d}t} = \omega_{\mathrm{m}}(t) \tag{2.24c}$$

$$\theta(t) = \frac{1}{N_{\mathrm{GB}}}\theta_{\mathrm{m}}(t) \tag{2.24d}$$

$$\omega(t) = \frac{1}{N_{\mathrm{GB}}}\omega_{\mathrm{m}}(t). \tag{2.24e}$$

This is clearly a nonlinear differential algebraic equation, which can easily be converted to an ordinary differential equation by inserting (2.24d) and (2.24e) where appropriate.

2.1.3 A model of the closed loop system

The system is controlled by a PI controller. In this section the model derived in Section 2.1.2 will be extended to include the PI controller. We will then have a model of the closed loop system, relating the speed reference and the motor speed. In the real system there is also a current controller in parallel with the speed controller that limits the motor voltage if too much current is drawn from the circuit. The current controller will not be included in the simulation model.

With the speed reference defined as $r(t)$, the integrated error as $E(t)$ and the proportional and integral gain of the PI controller as K_p and K_i, the closed loop model can be formulated as

$$L_{\mathrm{m}}\frac{\mathrm{d}i(t)}{dt} = u_{\mathrm{A}}(t) - R_{\mathrm{m}}i(t) - \psi\omega_{\mathrm{m}}(t) \tag{2.25a}$$

$$J_{\mathrm{tot}}(\theta(t))\frac{\mathrm{d}\omega_{\mathrm{m}}(t)}{\mathrm{d}t} = \psi i(t) + \tau_{\mathrm{mf}}(\omega(t)) + \tau_{\mathrm{wf}}(\theta(t),\omega(t)) + \tau_{\mathrm{g}}(\theta(t)) \tag{2.25b}$$

$$\frac{\mathrm{d}\theta_{\mathrm{m}}(t)}{\mathrm{d}t} = \omega_{\mathrm{m}}(t) \tag{2.25c}$$

$$\theta(t) = \frac{1}{N_{\mathrm{GB}}}\theta_{\mathrm{m}}(t) \tag{2.25d}$$

$$\omega(t) = \frac{1}{N_{\mathrm{GB}}}\omega_{\mathrm{m}}(t) \tag{2.25e}$$

$$\frac{\mathrm{d}E(t)}{\mathrm{d}t} = r(t) - \omega_{\mathrm{m}}(t) \tag{2.25f}$$

$$u_{\mathrm{A}}(t) = K_p(r(t) - \omega_{\mathrm{m}}(t)) + K_iE(t). \tag{2.25g}$$

2.1.4 A linear model of the closed loop system

The model derived in Section 2.1.2 is useful for simulation. However, for controller design a simpler model is useful. The model (2.25) contains four derivatives, but

since L_m is small, (2.25a) can be reduced to an algebraic equation. Furthermore, since $\theta(t)$ only appears in the non-linearities, (2.25c) is not needed. This leaves a second order system. Therefore, a second order discrete time transfer function will now be defined. The model was found using the system identification toolbox in Matlab using step response data from the test rig. The input to the model is the reference motor speed and the output is motor speed

$$\omega_\mathrm{m} = \hat{G}\omega_r. \tag{2.26}$$

The identified model is

$$\hat{G}(\mathrm{q}) = \frac{0.08249\mathrm{q}^{-1} - 0.0817\mathrm{q}^{-2}}{1 - 1.88\mathrm{q}^{-1} + 0.8807\mathrm{q}^{-2}}, \tag{2.27}$$

where q is the time shift operator, i.e. $x[t+i] = q^i x[t]$. The sampling time is 0.01 s.

2.1.5 Model verification

In this section the models derived in sections 2.1.2, 2.1.3 and 2.1.4 are verified. A step response is simulated in Simulink for each model and then the results are compared to measurements from the test rig. The model parameters used in the simulations are given in Appendix A.

First, the open-loop model from Section 2.1.2 is verified. A 24 V step in motor voltage u_A was applied to both the test rig and the simulation model. The results from the simulation and the experiment are presented in Figure 2.4. As can be seen in the figure, the step responses match reasonably well. The rise time is similar and the main part of the nonlinear behavior is captured.

Next, the closed-loop model from Section 2.1.3 and the linear model from Section 2.1.4 are verified. Here, a step in reference motor speed from zero to 509 rad/s was applied at time $t = 1$. This motor speed corresponds to a wiping frequency of 1 Hz. The simulation results and the measurements from the experiment in the test rig are presented in Figure 2.5. The nonlinear model is not perfect but captures most of the dynamics of the real system. One can see that the oscillations are smaller in the simulated system than in the measurements from the real system. A reason for this may be that a simplified speed controller is used in the simulation and that the current controller in parallel with the speed controller is omitted. Further, the simulation model does not include the speed estimator or any time delays.

From Figure 2.5 one can see that the linear model has the correct rise time but does of course not capture any of the nonlinear properties of the system.

The models derived in this chapter do not match the real system exactly. However, the models are judged to be good enough for evaluation of ILC algorithms.

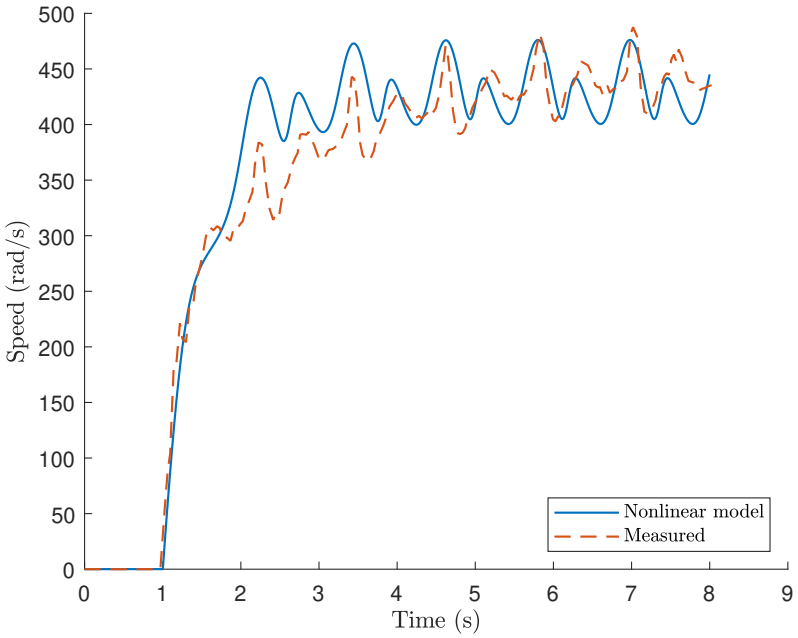


Figure 2.4: Simulated and measured response to a 24V step in motor voltage at time $t = 1$.

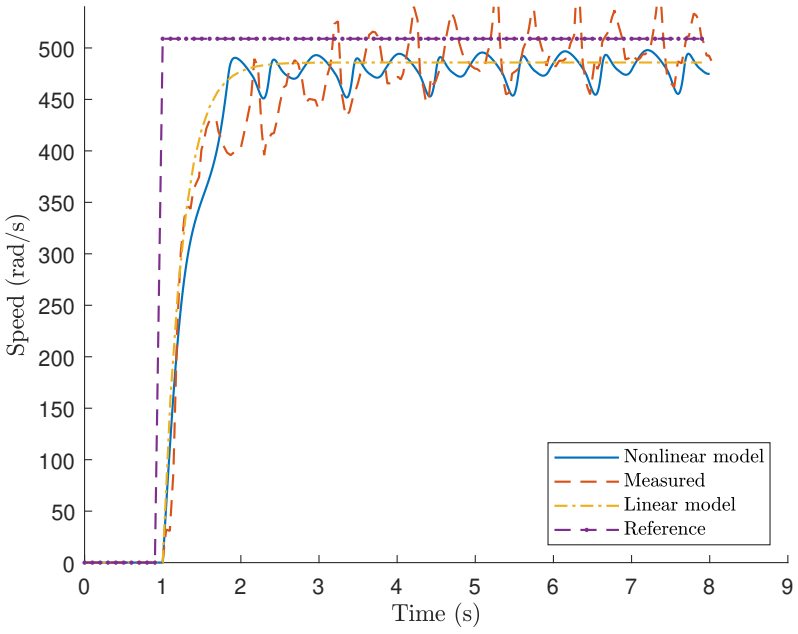


Figure 2.5: Simulated step response of the closed loop system. A step and the measured step responses are shown.

2.2 Choosing a non-clashing trajectory

In order to avoid a clash and analyze when it occurs, we need to quantify what a clash is. This can be considered a purely geometric problem at any moment in time. With the points $P_i \in \mathbb{R}^2$ defined as in Figure 2.6, a clash occurs when the signed distance between point P_1 and the line through the points P_2 and P_3 is positive, i.e. when the points P_1, P_2 and P_3 form a triangle to the right. From linear algebra we know that the signed distance in two dimensions is proportional to

$$d(\theta_1, \theta_2) \propto \det \begin{bmatrix} 1 & P_2^\mathsf{T}(\theta_1) \\ 1 & P_3^\mathsf{T}(\theta_1) \\ 1 & P_1^\mathsf{T}(\theta_2) \end{bmatrix}. \tag{2.28}$$

Then, the two wipers clash if

$$d(\theta_1, \theta_2) > 0. \tag{2.29}$$

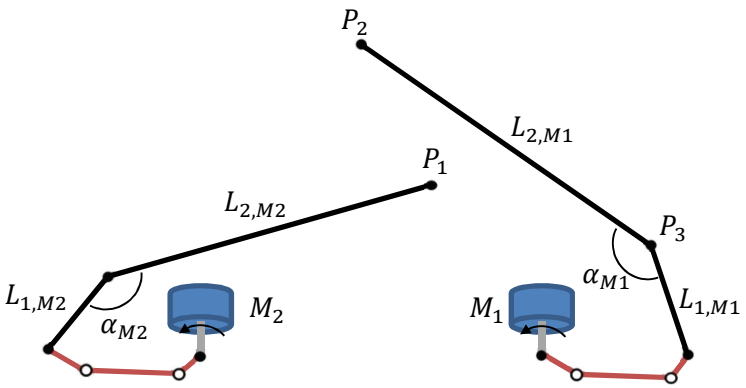


Figure 2.6: Schematic image of the wiper system with the points P_1, P_2 and P_3 defined. The parameters $L_{1,\mathrm{M1}}$, $L_{2,\mathrm{M1}}$, $L_{1,\mathrm{M2}}$, $L_{2,\mathrm{M2}}$, α_{M1} and α_{M2} are defined in Appendix A.

2.2.1 Angular trajectory generation

Calculating if the two wipers clash using (2.29) over all possible wiper configurations on a rotation gives the plot found in Figure 2.7. The goal can be seen as to get from $(0, 0)$ to $(2\pi, 2\pi)$ while keeping $d < 0$ and keeping θ_1 and θ_2 non-decreasing. By inspection it can be seen that a straight line, i.e. both motor having the same speed, does not work since the margin to a clash then would be too small. Intuitively, an S-shaped curve is preferable. We choose a modified sigmoid function which we will now define. The original sigmoid function is

$$\sigma_0(x) = \frac{1}{1 + \mathrm{e}^{-b(x-m)}} \tag{2.30}$$

where b and m are the steepness and midpoint, respectively.

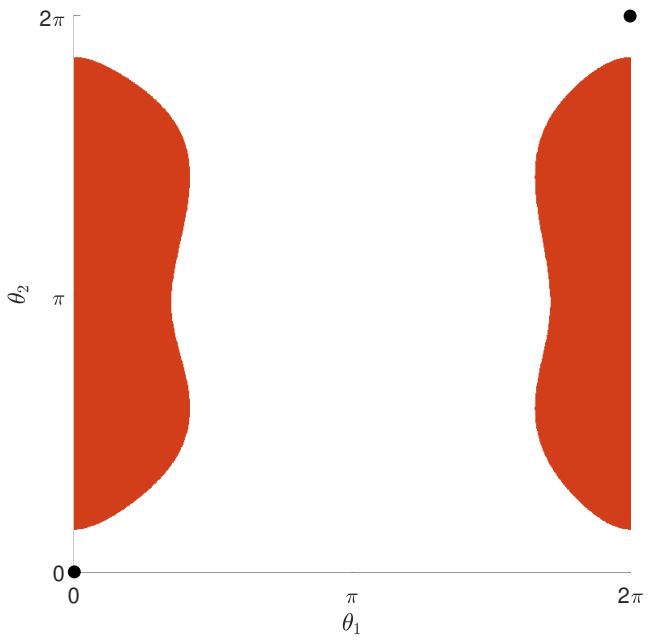


Figure 2.7: Motor angles θ_1 and θ_2 for the two wiper motors and the areas where the wipers clash. The area where the wipers do not clash ($d < 0$) is white.

The sigmoid goes asymptotically towards zero and one, as it is usually used within statistics. However, we wish that it should cross the origin and $(2\pi, 2\pi)$. Thus, we introduce the modified sigmoid

$$\sigma(x) = \frac{a}{1 + \mathrm{e}^{-b(x-m)}} + c\,. \tag{2.31}$$

The boundary conditions

$$\begin{cases} \sigma(0) = 0 \\ \sigma(2\pi) = 2\pi \end{cases} \quad \Longrightarrow \quad \begin{cases} a = \dfrac{2\,\pi\,\left(\mathrm{e}^{b\,(m-2\,\pi)} + 1\right)\,\left(\mathrm{e}^{b\,m} + 1\right)}{\mathrm{e}^{b\,m} - \mathrm{e}^{b\,(m-2\,\pi)}} \\ c = \dfrac{-a}{1 + \mathrm{e}^{bm}} \end{cases}. \tag{2.32}$$

This leaves $b > 0$ and $m \in [0, 2\pi]$ as design variables. We choose $m = \pi$ for symmetry. Letting $b \to 0$ gives a straight line and $b \to \infty$ gives a step at m.

The shape of the area where the wipers clash in Figure 2.7 depends on the model parameters for the linkage and the wiper arms. Thus, for different bus models, the areas will be different. If the parameters are uncertain, or if the generated reference profile should be useful for multiple bus models, the curve must be chosen such that there is some margin to the clash areas.

In Figure 2.8 we can see four generated paths with different values of b. From the figure it is clear that if b is chosen too small, the wipers are likely to clash since the margin to the clash area is small. On the other hand, if b is chosen unnecessarily large, the motors will have to accelerate more than necessary. From inspection, $b = 0.7$ seems to be a good compromise between clash margin and constant speed.

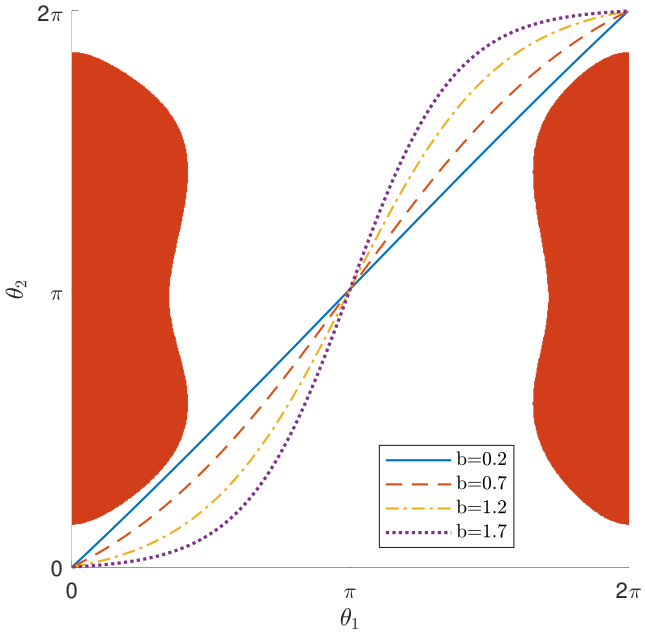


Figure 2.8: Generated paths for four different values of b and the areas where the wipers clash.

2.2.2 Velocity trajectory generation

While the trajectory generated in Section 2.2.1 gives a clash free trajectory and the relative velocity between the motors, it does not provide any absolute velocities. Thus, we now want to choose velocity trajectories, such that the motor angles follow their trajectories and that the cycle time is achieved. To do this, we want to create $\theta_1(t)$ and $\theta_2(t)$ instead of the parametric form $\theta_2(\theta_1) = \sigma(\theta_1)$.

A method for moving along parametric curves with an arbitrary speed profile is presented in [5]. Parts of the method will be repeated here for convenience, but with notation relevant to this work.

Consider the parametric curve that relates θ_1 to θ_2

$$\mathbf{Y}(\theta_1) = \begin{bmatrix} \sigma(\theta_1) \\ \theta_1 \end{bmatrix} \tag{2.33}$$

over $\theta_1 \in [\underline{\theta}, \overline{\theta}] = [0, 2\pi]$, where σ is defined in (2.31). We also have the yet to be determined time-parameterized curve

$$\mathbf{X}(t) = \begin{bmatrix} \theta_1(t) \\ \theta_2(t) \end{bmatrix} . \tag{2.34}$$

We want to determine $\mathbf{X}(t)$ such that the speed along the parametric curve, $|\mathrm{d}\mathbf{X}/\mathrm{d}t|$, at time $t \in [\underline{t}, \overline{t}]$, is specified by a function $\Omega(t)$. By choosing a function $\bar{\Omega}(t)$ with the desired shape, the function $\Omega(t)$ can be obtained from

$$\Omega(t) = \frac{L\bar{\Omega}(t)}{\int_{\underline{t}}^{\overline{t}} \bar{\Omega}(t)\mathrm{d}t} , \tag{2.35}$$

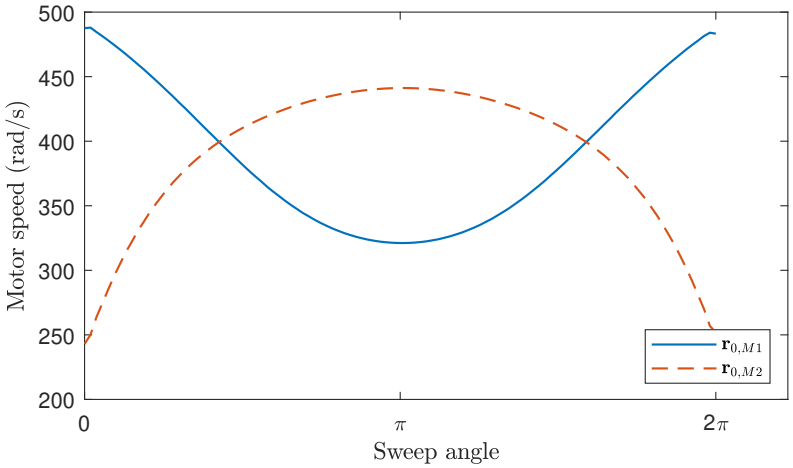


Figure 2.9: Calculated references $\mathbf{r}_{0,\mathrm{M1}}$ and $\mathbf{r}_{0,\mathrm{M2}}$ where $\Omega(t)$ is constant, $b = 0.7$ and the cycle time is 1.33 s.

where the curve length L is defined as

$$L = \int_{\underline{\theta}}^{\overline{\theta}} \left| \frac{\mathrm{d}\mathbf{Y}}{\mathrm{d}\theta_1} \right| \mathrm{d}\theta_1 = \int_{\underline{t}}^{\overline{t}} \left| \frac{\mathrm{d}\mathbf{X}}{\mathrm{d}t} \right| \mathrm{d}t. \tag{2.36}$$

This ensures that the desired cycle time is achieved.

Now, the distance traveled is known as

$$\ell = \int_{\underline{t}}^{T} \Omega(t) \mathrm{d}t \tag{2.37}$$

with the equivalent expression

$$\ell = \int_{\underline{\theta}}^{\theta_1} \left| \frac{\mathrm{d}\mathbf{Y}}{\mathrm{d}\tilde{\theta}_1} \right| \mathrm{d}\tilde{\theta}_1 \,. \tag{2.38}$$

Combining (2.37) and (2.38) yields $t(\theta_1)$, which can be inverted numerically with Newton's method. Pseudocode for this procedure is given in [5]. The procedure can give an arbitrarily fine trajectory of $\theta_1(t)$ for $t \in [\underline{t}, \overline{t}]$. Then $\theta_2(t)$ is given from the relationship derived in Section 2.2.1. The velocity references $\mathbf{r}_{0,\mathrm{M1}}$ and $\mathbf{r}_{0,\mathrm{M2}}$ can be calculated by numerically differentiating $\theta_1(t)$ and $\theta_2(t)$. An example of calculated references is presented in Figure 2.9.

3

Iterative Learning Control

In this chapter we will start by introducing the concept of iterative learning control (ILC) in a general form and then proceed with the investigated algorithms. An excellent overview and introduction to ILC can also be found in [6] and [7].

3.1 Introduction to iterative learning control

Iterative learning control is a control strategy that can be utilized for iterative systems. That means that the same, or similar, operations should be performed many times under the same conditions. This makes it especially suitable for robot control for manufacturing and other mass production related processes, where the same action is repeated [6]. It has successfully been implemented in robot systems [8], induction motor control [9] and chain conveyor systems [10].

ILC can be applied either on its own, in parallel or in series with a standard feedback controller, where a serial architecture can be used when applying ILC to an existing control system [6]. This architecture is shown in Figure 3.1. The closed loop system formed by C and P will be considered fixed for this thesis.

A discrete time closed loop system is given on the form

$$\begin{aligned} x[t+1] &= f(x[t], r[t]) + v[t] \\ z[t] &= h(x[t]) \\ y[t] &= z[t] + w[t] \end{aligned} \tag{3.1}$$

where $x[t]$ is the state, $r[t]$ is the reference, $z[t]$ is the true output, $y[t]$ is the measured output, $v[t]$ is process noise and $w[t]$ is measurement noise. Then we denote the sequence of N samples of states, references, outputs and measurements over the interval $t \in [0, T]$ as

$$\begin{aligned} \mathbf{x}_k &= \begin{bmatrix} x_k[0] & \dots & x_k[N-1] \end{bmatrix}^\mathsf{T} \\ \mathbf{r}_k &= \begin{bmatrix} r_k[0] & \dots & r_k[N-1] \end{bmatrix}^\mathsf{T} \\ \mathbf{z}_k &= \begin{bmatrix} z_k[0] & \dots & z_k[N-1] \end{bmatrix}^\mathsf{T} \\ \mathbf{y}_k &= \begin{bmatrix} y_k[0] & \dots & y_k[N-1] \end{bmatrix}^\mathsf{T}, \end{aligned} \tag{3.2}$$

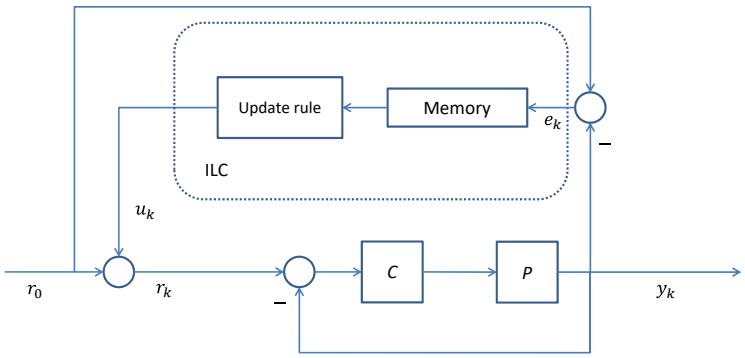


Figure 3.1: A serial architecture of the ILC algorithm, with the feedback controller C and plant P.

where the subscript k indicates the iteration number. From these, the error sequence $\mathbf{e}_k = \mathbf{r}_k - \mathbf{y}_k$ can also be constructed.

The concept of iterative learning control is to, for each iteration k, calculate a correction sequence $\mathbf{u}_k$ such that the output sequence $\mathbf{y}_k$ better follows $\mathbf{r}_0$. For each iteration, the correction signal is updated based on the measured output sequence from the previous iteration and the reference sequence $\mathbf{r}_0$ according to

$$\mathbf{u}_{k+1} = \mu(\mathbf{u}_k, \mathbf{y}_k, \mathbf{r}_0). \tag{3.3}$$

When applied in a serial architecture, the reference to the closed loop system for the next iteration is then calculated as

$$\mathbf{r}_{k+1} = \mathbf{r}_0 + \mathbf{u}_{k+1}. \tag{3.4}$$

The ILC approach differs from standard feedback control since the entire error sequence is available for calculations, enabling techniques such as acausal filtering. The update rule in (3.3) is a first order ILC algorithm since $\mathbf{u}_{k+1}$ is calculated using data only from the previous iteration. A higher order ILC algorithm uses information from multiple iterations. Higher order ILC algorithms are investigated in [7] where the benefit over first order algorithms was shown to be small. Thus, higher order ILC algorithms will not be investigated in this work.

The goal of ILC algorithm design is to choose the function μ such that the iterative system is well-behaved. In order to analyze and compare different ILC algorithms it is necessary to first define what a well-behaved system is. In control system design, stability is often a primary concern. In the ILC field, stability is often called convergence instead. A desired property is then to have monotonic convergence, i.e. the error $\mathbf{e}_k$ never increases between iterations, often in a L2-norm sense. Closely related to this is the convergence speed, i.e. how many iterations are needed for convergence, and the asymptotic error, i.e. the error to which the algorithm converges.

If f, h and μ are linear functions, then the algorithm is said to be a linear ILC algorithm. The theory for linear ILC follows from linear systems theory and is thus well developed and understood.

If the system is linear, the system could also be written on standard state-space form

$$\begin{aligned} x_k[t+1] &= Ax_k[t] + Br_k[t] \\ y_k[t] &= Cx_k[t]. \end{aligned} \tag{3.5}$$

Furthermore, for a discrete time linear time-invariant single-input single-output system $G(\mathrm{q})$, the general system description in (3.1) can be reformulated as

$$y_k[t] = G(q)r_k[t], \tag{3.6}$$

where q is the time shift operator.

3.1.1 Lifted notation

To simplify the analysis of linear ILC algorithms, the system is often described in so-called lifted notation, or matrix form. Denote the impulse response of $G(\mathrm{q})$ as $g[t]$, i.e.

$$G(\mathrm{q}) = T_s \sum_{i=0}^{\infty} g[i]\mathrm{q}^{-i}, \tag{3.7}$$

where T_s is the sample time. Then it is possible to form the matrix

$$\mathbf{G} = T_s \begin{bmatrix} g[0] & 0 & \dots & 0 \\ g[1] & g[0] & & \vdots \\ \vdots & & \ddots & 0 \\ g[N-1] & g[N-2] & \dots & g[0] \end{bmatrix}. \tag{3.8}$$

This gives the system notation

$$\mathbf{y}_k = \mathbf{G}\mathbf{r}_k. \tag{3.9}$$

Here, the full sequence of outputs is calculated from the full sequence of inputs for iteration k.

3.1.2 Assumptions in Iterative Learning Control

The theory of iterative learning control has been developed under the following assumptions, from [7] and [11]. They are rephrased here to fit the notation of this work.

1. Every iteration ends in a fixed time of duration $T > 0$.
2. A desired output $r_0[t]$ is given a priori over that time with duration $t \in [0, T]$.
3. Repetition of the initial setting is satisfied, that is, the initial state $x_k[0]$ of the objective system can be set the same at the beginning of each trial:

$$x_k[0] = x_0[0],\ k \in \mathbb{Z}^+ \tag{3.10}$$

4. Invariance of the system dynamics is ensured throughout these repeated iterations.
5. Every output $z_k[t]$ can be measured and therefore the tracking error signal, $e_k(t) = r_0[t] - z_k[t]$, can be utilized in the calculation of $u_{k+1}[t]$.
6. Given a reference trajectory $r_0[t], t \in [0, T]$, with a piecewise continuous derivative, there exists a unique input trajectory $r^*[t]$ on the same time interval such that $z[t]$ equals $r_0[t]$.

The assumptions 3 and 5 are generally considered too strict and not practical for real systems. As such, they are generally relaxed as in [7, p. 28-29] to:

3a. The system is initialized at the beginning of every iteration such that the error in the initial state is limited. This means that if the controlled system is written in state space form the initial state $x_k[0]$ fulfills

$$|x_k[0] - x_0[0]| < \epsilon$$

for some $\epsilon > 0$.

5a. It is possible to measure $y_k[t] = z_k[t] + w_k[t]$, where $w_k[t]$ is a measurement disturbance.

If measurements of the states are not available, assumption 5a can be relaxed even further with the introduction of a state-observer, as investigated in [12]. This will however not be used in this thesis. Implications of these assumptions will be discussed further in Chapter 6.

3.2 Heuristic Iterative Learning Control

One of the earliest and simplest ILC algorithms is a model-free algorithm that is referred to in [7] as heuristic ILC. The algorithm consists of two filters and the update is given as

$$u_{k+1}[t] = Q(q)(u_k[t] + L(q)(r[t] - y_k[t]))\,. \tag{3.11}$$

The filter L is often an acausal filter, $L = \kappa q^n$ where κ is a gain and q^n is a time shift of n steps. The filter Q is either chosen as unity or a zero-phase low-pass filter to increase robustness. The zero-phase property can be achieved with functions such as `filtfilt` in Matlab.

It is well known that robustness is in conflict with performance. Thus $Q = 1$ gives the best nominal performance, but leaves the system susceptible to disturbances. Often, Q is chosen as $Q = \bar{Q}(q)\bar{Q}(q^{-1})$, where $\bar{Q}(q)$ is a low pass filter. This choice of filter will increase robustness, but gives a non-zero asymptotic error [7].

A sufficient frequency domain condition for convergence of the algorithm is derived in [7]. The algorithm is guaranteed to converge if

$$|1 - L(e^{j\omega})G(e^{j\omega})| < |Q^{-1}(e^{j\omega})|\,,\ \forall\omega. \tag{3.12}$$

3.3 Norm optimal Iterative Learning Control

This method is an optimal control approach to the ILC design, similar to linear quadratic regulator design. The algorithm described here is based on [13] and [14] and is adopted to the notation from Section 3.1 and 3.1.1. For each new iteration we want to find the control sequence $\mathbf{u}_{k+1}$ that minimizes the following quadratic cost function

$$J_{k+1} = \mathbf{e}_{k+1}^{\mathsf{T}} \mathbf{W}_e \mathbf{e}_{k+1} + \mathbf{u}_{k+1}^{\mathsf{T}} \mathbf{W}_u \mathbf{u}_{k+1} + (\mathbf{u}_{k+1} - \mathbf{u}_k)^{\mathsf{T}} \mathbf{W}_{\Delta u} (\mathbf{u}_{k+1} - \mathbf{u}_k). \tag{3.13}$$

Here, $\mathbf{e}_{k+1} = \mathbf{r}_0 - \mathbf{y}_{k+1}$. The weighting matrices $\mathbf{W}_e \succ 0$, $\mathbf{W}_u \succeq 0$ and $\mathbf{W}_{\Delta u} \succeq 0$ penalize the output error, the input energy and the change of input between iterations, respectively.

The resulting optimal update equation is derived in [14] and is given by

$$\mathbf{u}_{k+1} = (\mathbf{W}_u + \mathbf{W}_{\Delta u} + \mathbf{G}^{\mathsf{T}} \mathbf{W}_e \mathbf{G})^{-1} ((\mathbf{W}_{\Delta u} + \mathbf{G}^{\mathsf{T}} \mathbf{W}_e \mathbf{G}) \mathbf{u}_k + \mathbf{G}^{\mathsf{T}} \mathbf{W}_e \mathbf{e}_k). \tag{3.14}$$

This can be expressed similar to Q- and L formulation in (3.11) as

$$\mathbf{u}_{k+1} = \mathbf{Q}(\mathbf{u_k} + \mathbf{L}\mathbf{e_k}) \tag{3.15}$$

with

$$\begin{aligned} \mathbf{Q} &= (\mathbf{W}_u + \mathbf{W}_{\Delta u} + \mathbf{G}^{\mathsf{T}} \mathbf{W}_e \mathbf{G})^{-1} (\mathbf{W}_{\Delta u} + \mathbf{G}^{\mathsf{T}} \mathbf{W}_e \mathbf{G}) \\ \mathbf{L} &= (\mathbf{W}_{\Delta u} + \mathbf{G}^{\mathsf{T}} \mathbf{W}_e \mathbf{G})^{-1} \mathbf{G}^{\mathsf{T}} \mathbf{W}_e. \end{aligned} \tag{3.16}$$

Conditions for convergence of the algorithm are given in [7]. For the system (3.9) controlled by the ILC update rule in (3.15), monotonic exponential convergence is achieved if

$$\bar{\sigma}(\mathbf{Q}(\mathbf{I} - \mathbf{L}\mathbf{G})) < 1\,, \tag{3.17}$$

where $\bar{\sigma}(M)$ denotes the largest singular value of the matrix M.

It should be noted here that the matrices in (3.14) have dimension $N \times N$. So, although the lifted notation is convenient for analysis of the algorithm and simulation on computer systems where memory is not a problem, it is not practical for implementations on a microcontroller. Therefore, two versions of norm optimal ILC that are easier to implement in practice will now be investigated.

3.3.1 A filter implementation of Norm optimal Iterative Learning Control

A filter implementation of norm optimal ILC will now be described. This algorithm is based on the algorithm presented in Section 3.3 but uses a frequency domain representation and acausal filters instead of the matrix formulation. A detailed derivation of the algorithm is given in [14] and a summary will be presented here.

With $\mathbf{W}_u = \rho \cdot I$, $\mathbf{W}_{\Delta u} = \lambda \cdot I$ and $\mathbf{W}_e = I$ (3.16) becomes

$$\begin{aligned} \mathbf{Q} &= (\rho \cdot I + \lambda \cdot I + \mathbf{G}^{\mathsf{T}} \mathbf{G})^{-1} (\lambda \cdot I + \mathbf{G}^{\mathsf{T}} \mathbf{G}) \\ \mathbf{L} &= (\lambda \cdot I + \mathbf{G}^{\mathsf{T}} \mathbf{G})^{-1} \mathbf{G}^{\mathsf{T}}. \end{aligned} \tag{3.18}$$

It is then possible to utilize that $\mathbf{y} = \mathbf{G}^{-1}\mathbf{u}$ corresponds to the filtering operation $y[t] = \frac{1}{G(q)}u[t]$ and similarly that $\mathbf{y} = \mathbf{G}^{\mathsf{T}}\mathbf{u}$ corresponds to the acausal filtering operation $y[t] = G(q^{-1})u[t]$. Now it is easy to see that the corresponding filtering operations to (3.18) are

$$\begin{aligned} Q(q) &= \frac{\lambda + G(q)G(q^{-1})}{\rho + \lambda + G(q)G(q^{-1})} \\ L(q) &= \frac{G(q^{-1})}{\lambda + G(q)G(q^{-1})}. \end{aligned} \tag{3.19}$$

As pointed out in [14] the filtering representation in (3.19) is only an approximation of (3.18) since boundary effects are neglected when the filters are applied to a finite sequence of N samples.

To implement the acausal filters $Q(q)$ and $L(q)$ it is first necessary to find $\bar{Q}(q)$ and $\bar{L}(q)$ such that $Q = \bar{Q}(q)\bar{Q}(q^{-1})$ and $L = \bar{L}(q)\bar{L}(q^{-1})$. Then, the forward and backward filtering can be performed using the same technique as in the heuristic algorithm described in Section 3.2. To analyze the stability of the iterative system (3.12) can be used.

3.3.2 A recursive implementation of Norm optimal Iterative Learning Control

Another norm optimal algorithm is proposed in [15, pp. 238-240] for systems on state-space form. It intends to minimize the cost criteria (3.13) but with $\mathbf{W}_u = 0$. Note that the weighting matrices $\mathbf{W}_e$ and $\mathbf{W}_{\Delta u}$ could be chosen to be time variant.

The algorithm uses time varying state feedback matrices $K[t]$ and $\tilde{K}[t]$ that can be computed a priori as

$$\begin{aligned} \tilde{K}[t+1] &= A^{\mathsf{T}}K[t+1] + C^{\mathsf{T}}\mathbf{W}_e[t+1]C \\ K[t] &= \left(I + \tilde{K}[t+1]B\mathbf{W}_{\Delta u}^{-1}[t]B^{\mathsf{T}}\right)^{-1}\tilde{K}[t+1]A \end{aligned} \tag{3.20}$$

for $t \in [0, N-1]$ where $K[N-1] = 0$ is the terminal condition of a reversed recursive calculation. The predictive feedforward term $\xi_{k+1}[t]$ and ILC update $u_{k+1}[t]$ are then calculated between iterations as

$$\begin{aligned} \xi_{k+1}[t] &= \left(I + \tilde{K}[t]B\mathbf{W}_{\Delta u}^{-1}[t]B^{\mathsf{T}}\right)^{-1} \\ &\quad \left(A^{\mathsf{T}}\xi_{k+1}[t+1] + C^{\mathsf{T}}\mathbf{W}_e[t+1]e_k[t+1]\right),\ t \in [0, N-1] \\ \xi_{k+1}[N] &= 0 \end{aligned} \tag{3.21}$$

and

$$u_{k+1}[t] = u_k[t] + \mathbf{W}_{\Delta u}^{-1}[t]B^{\mathsf{T}}\Big(- K[t](x_{k+1}[t] - x_k[t]) + \xi_{k+1}[t]\Big) \tag{3.22}$$

for $t \in [0, N-1]$ where again $\xi_{k+1}[t]$ is a backwards recursive calculation. It should be noted that (3.22) assumes that all states are available for feedback. If not all states are measured, a state observer can be used [13].

The standard version of the algorithm is a so-called current iteration ILC algorithm, i.e. it uses feedback from the current iteration in the calculation of u_{k+1}. This scheme is equivalent to using an internal feedback controller [6, p. 99]. Since a serial ILC architecture is used in the wiper system, the term $(x_{k+1}[t] - x_k[t])$ will be set to zero to avoid introducing additional feedback. This makes the algorithm purely feedforward as fits with the framework of the system. This also removes the need for a state observer for higher order systems.

The formal criteria for convergence is given in [13] and [15, th. 9.2], but for well behaved 'regular' systems with small modeling errors, exponential rate of convergence is guaranteed for both error norm and input norm.

3.4 RoFaLT

The Robust and Fast Learning Tool (RoFaLT) is a toolbox developed in Matlab for nonlinear ILC for nonlinear systems [16]. It is a two step optimization-based algorithm that both improves a candidate model and calculates the optimal control for this model. Both of these steps are formulated in the form of nonlinear programming (NLP) problems, that are solved after each iteration.

If the system is defined as in (3.1), the first NLP that solves the model correction problem is given by

$$\min_{x,\alpha} \quad \frac{1}{2}\sum_{t=0}^{N-1}\left[\|y_k[t] - h\left(x[t], r_k[t], \alpha[t]\right)\|_2^2 + \|\alpha[t]\|_{\mathbf{R}_1}^2 + \|\Delta_i\alpha[t]\|_{\mathbf{W}_1}^2\right] + \frac{1}{2}\sum_{t=0}^{N-2}\|\Delta_t\alpha[t]\|_{\mathbf{W}_2}^2 \tag{3.23a}$$

$$\begin{aligned} \text{s.t.} \quad x[t+1] &= f\left(x[t], r_k[t], \alpha[t]\right) \\ r_k[t] &= r_0[t] + u_k[t] \\ \Delta_t\alpha[t] &= \alpha[t+1] - \alpha[t], \qquad t = 0, \ldots, N-2 \\ \Delta_i\alpha[t] &= \alpha[t] - \alpha_k^\star[t] \\ \underline{\alpha} \leq \alpha[t] &\leq \overline{\alpha}, \qquad t = 0, \ldots, N-1 \end{aligned} \tag{3.23b}$$

where (3.23a) is a weighted-norm minimization problem with the regularization weight matrices $\mathbf{R}_1$, $\mathbf{W}_1$ and $\mathbf{W}_2$. The regularization of the variables defined in (3.23b) can be seen as low-pass filtering in the time and iteration domain, respectively. The solution to (3.23), $\alpha_k^\star$, is then used in the control problem, defined as

$$\min_{x,u} \quad \frac{1}{2}\sum_{t=0}^{N-1}\left[\|r_0[t]-h\left(x[t],r_k[t],\alpha_k^\star[t]\right)\|_2^2+\|u[t]\|_{\mathbf{R}_2}^2+\|\Delta_i u[t]\|_{\mathbf{W}_3}^2\right]+ \\ +\frac{1}{2}\sum_{t=0}^{N-2}\|\Delta_t u[t]\|_{\mathbf{W}_4}^2 \tag{3.24a}$$

$$\begin{aligned} \text{s.t.} \quad x[t+1] &= f\left(x[t],r_k[t],\alpha_k^\star[t]\right) \\ r_k[t] &= r_0[t]+u[t] \\ \Delta_t u[t] &= u[t+1]-u[t], \qquad t=0,\ldots,N-2 \\ \Delta_i u[t] &= u[t]-u_k^\star[t] \\ \underline{u} &\le u[t] \le \overline{u}, \qquad t=0,\ldots,N-1 \end{aligned} \tag{3.24b}$$

where the regularization matrices $\mathbf{R}_2$, $\mathbf{W}_3$ and $\mathbf{W}_4$ has similar function as above.

When modeling the system for use with RoFaLT, one must include where and how model corrections can be made. This is done in the form of the parametric and non-parametric corrections terms $\alpha_k[t]$. The difference between them is that parametric terms are kept constant over an iteration, while the nonparametric terms are time dependent.

The NLP formulation of the ILC problem allows for the addition of other objectives, such as energy consumption. This makes the RoFaLT algorithm very flexible and powerful, however also very complex given that one can add more design parameters. This could potentially make tuning difficult.

It should be noted that if a linear model is used with additive nonparametric α_k corrections on the output, we obtain the same optimization problem as in (3.13), only not with a closed form update equation.

3.5 Using iterative learning control with a limited number of setpoints

In the wiper control system the speed references are specified using eight setpoints. The reference speed at a specific motor angle is then calculated using linear interpolation.

So far the assumption has been that the ILC correction $\mathbf{u}_k$ and the ILC reference $\mathbf{r}_k$ is of length N, the same size as the measurement sequence $\mathbf{y}_k$. From a usability perspective it would be preferable if the ILC correction could be applied directly to the eight setpoints that specify the reference speed. However, N is typically much larger than eight. The trivial solution to this is to increase the sample time such that $N = 8$. This method has a number of problems. For instance, only eight measurements would be taken during a sweep. Then, the ILC algorithm would minimize the error only at the eight sample points whereas the goal is to minimize the error over the entire sweep, including between the eight setpoints.

We propose an alternative approach that uses measurements from the entire sweep to calculate ILC corrections at eight points. We propose to first calculate an ILC reference $\mathbf{r}_k$ of length N and then fit a continuous piecewise linear function to $\mathbf{r}_k$ using the least squares method. The fitted function will then be defined by values at eight breakpoints which correspond to the 8 setpoints in the controller. Between the breakpoints the function is linear which agrees exactly with how the reference is calculated in the controller. The least squares fit can be done using the method presented in [17], where a closed form solution is given to the optimal solution. As illustrated in Figure 1.2, the setpoint at sweep angle 2π is actually the first setpoint. Thus, we need to add a periodic constraint to the algorithm in [17] such that the ninth setpoint always is equal to the first. An implementation of this algorithm with added periodic constraint is available in [18]. The method is illustrated in Figure 3.2, for details of the algorithm we refer to [17] and [18]. The serial ILC architecture with the splinefit modification is shown in Figure 3.3.

The effect of the splinefit method on the algorithms' performance will be investigated in simulations in Chapter 4.

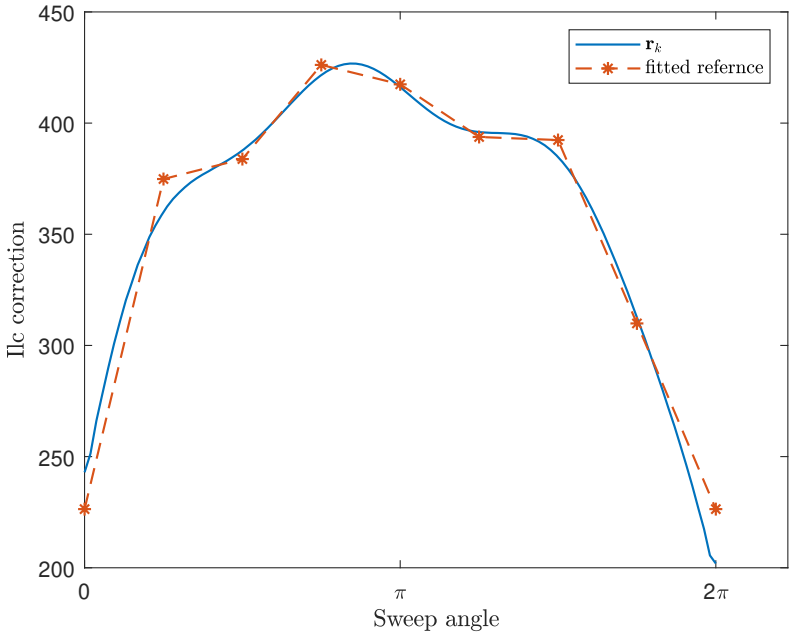


Figure 3.2: Illustration of the splinefit algorithm with periodic constraint. The dense ILC reference $\mathbf{r}_k$ and the fitted piecewise linear function are shown.

3.6 Sampling at constant angle intervals

In ILC literature, sampling is normally performed with constant sample time. It is also assumed that the number of samples is constant for all iterations. This implies that each iteration lasts for the same length of time. Since the ILC algorithm varies the speed of the wipers, it is clear that this method does not guarantee that a sample sequence contains samples from a full wiper sweep. If the wipers move too slow one

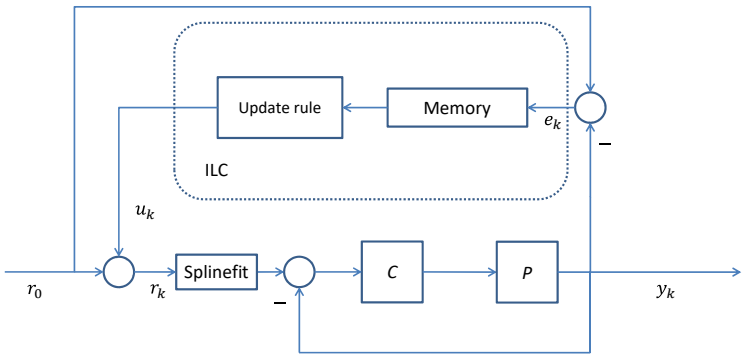


Figure 3.3: A serial architecture of the ILC algorithm, with the splinefit algorithm included.

iteration, the sample sequence will not contain samples from the last part of the sweep. If the wipers move too fast, the sample sequence would contain samples from the beginning of the next sweep. In the windshield wiper system it would be useful if the sampling instead could be performed at constant angle intervals. This would guarantee that each sample sequence contains samples from the full wiper sweep while the number of samples still is constant. Moreover, constant angle sampling simplifies the implementation and agrees well with how the reference speed is calculated by the inner controller (see Section 1.1.1 and Figure 1.2). The drawback of this method is that the interpretation of the filtering operations are less clear when motor angle has replaced time as the independent variable. Also, the model based algorithms are using models derived in the time domain, which will lose some interpretation as well. The effect of this modification to the algorithms is investigated in simulations in Chapter 4.

4

Evaluation of ILC algorithms

In this chapter we introduce the simulation environment that will be used to evaluate the different algorithms described in Chapter 3. Then we present results from simulations and discuss their implications.

4.1 Simulation setup

To simulate the system in closed loop with the ILC controller, the model presented in Section 2.1.3 was implemented in Simulink together with the reference generation procedure presented in Section 1.1.1. To evaluate the effects of the splinefit procedure, a continuous reference was also implemented which corresponds to how the ILC algorithms are implemented in the literature.

Following the assumptions made in Section 3.1.2, the angular velocity of the system was initialized to the reference value and a fixed number of samples were collected. For the time based sampling case, a sampling time of 0.01 s was used, yielding 134 samples for the required sweeping frequency. When using constant angle sampling, 108 samples were used, giving a resolution of 3.33° on the gearbox side.

The performance metric used is the L2-norm of the error vector $\mathbf{e}_k$. For illustration purposes, we normalize it with the zeroth iteration value as

$$L_{2,k} = \frac{\sqrt{\sum_{t=0}^{N} e_k^2[t]}}{\sqrt{\sum_{t=0}^{N} e_0^2[t]}}, \quad k \in \{0, 1, 2, 3, \ldots\}. \tag{4.1}$$

The algorithms work independently on the individual motor references. Because of this, only the results from one motor are presented in the following sections as the results for the second motor are analogous.

4.2 Algorithm tuning and stability

The algorithms have been tuned for fast convergence with some robustness to noise in mind. Due to the different tuning parameters, it is difficult to get comparable properties. However, convergence speed and monotonic convergence have been a priority where it has been possible. Since the system model is nonlinear, it cannot

be used with the available convergence criteria from the literature. Instead, the approximate model in (2.27) was used for stability analysis. Furthermore, the splinefit algorithm is not accounted for in the analysis.

The tuning used for the heuristic algorithm in (3.11) was chosen as $L = 0.8q^4$ and $\bar{Q}(q)$ as a second order Butterworth filter with normalized cutoff frequency $\omega_n = 0.2$ obtained from `butter(2, 0.2)` in Matlab. With this tuning the stability condition (3.12) is met.

Using the weight matrices defined in (3.18) for the matrix implementation, the weights were chosen as $\rho = 0.01$, $\lambda = 0.7$ and $G = \hat{G}$ from (2.27). This tuning fulfills (3.17). The filter implementation instead uses $\rho = 0.92$ to fulfill (3.12). The recursive algorithm can be parameterized similarly to (3.18) with $\rho = 0$. The tuning was chosen as $\lambda = 0.94$. The state space matrices A, B and C were calculated from $\hat{G}$ using the `ss` command in Matlab.

For the RoFaLT algorithm we used the model defined in (2.25), with multiplicative uncertainty added to b_w and J_ϕ through parametric α_k variables. Additionally, a non-parametric additive correction was added to the measurement. The bounds for α_k can be found in Table 4.1. The regularization matrices were chosen as

$$\begin{aligned} \mathbf{R}_1 &= 0.1 \cdot I \quad & \mathbf{W}_1 &= 0 \quad & \mathbf{W}_2 &= 0.01 \cdot I \\ \mathbf{R}_2 &= 10^{-6} \cdot I \quad & \mathbf{W}_3 &= 0 \quad & \mathbf{W}_4 &= 10^{-4} \cdot I. \end{aligned} \tag{4.2}$$

4.3 Simulation results

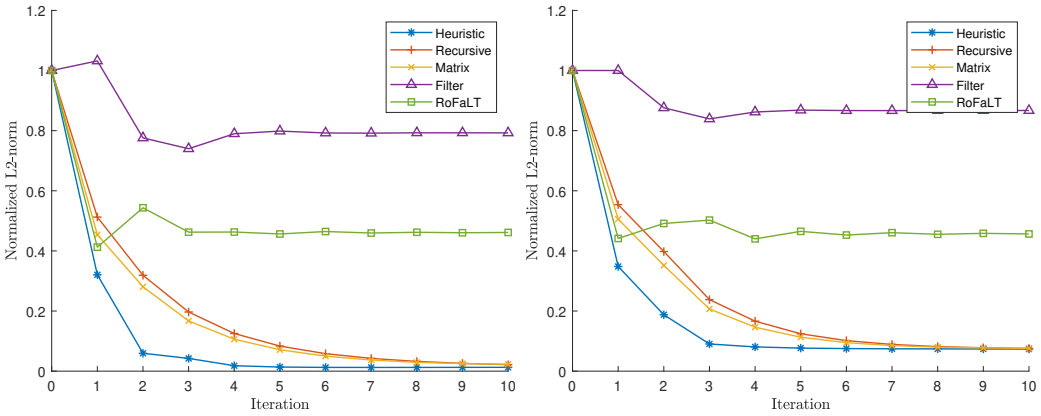
In Figure 4.1a we can see how the L2-norm of the tracking errors develop over iterations for the different algorithms. The figure shows that the performance is similar for the heuristic algorithm and two of the norm optimal ones. All these show exponential convergence. The filter implementation of norm optimal ILC and the RoFaLT algorithm also converge, but to worse asymptotic errors. The corresponding tracking curves after ten iterations can be found in Figure 4.2a.

4.3.1 The effects of splinefit

As we can see in Figure 4.1b, the splinefit procedure, presented in Section 3.5, affects the performance. It is clear that the splinefit algorithm reduces the performance of the ILC algorithms. The algorithms with splinefit cannot fully compensate for the curvature of the reference curve and the nonlinear dynamics and thus, the asymptotic

Table 4.1: Correction bounds for the correction parameters α_k used in the RoFaLT algorithm.

Variable	Min. correction	Max. correction
b_w	0.1	10
J_ϕ	0.1	10
y	$-\infty$	∞



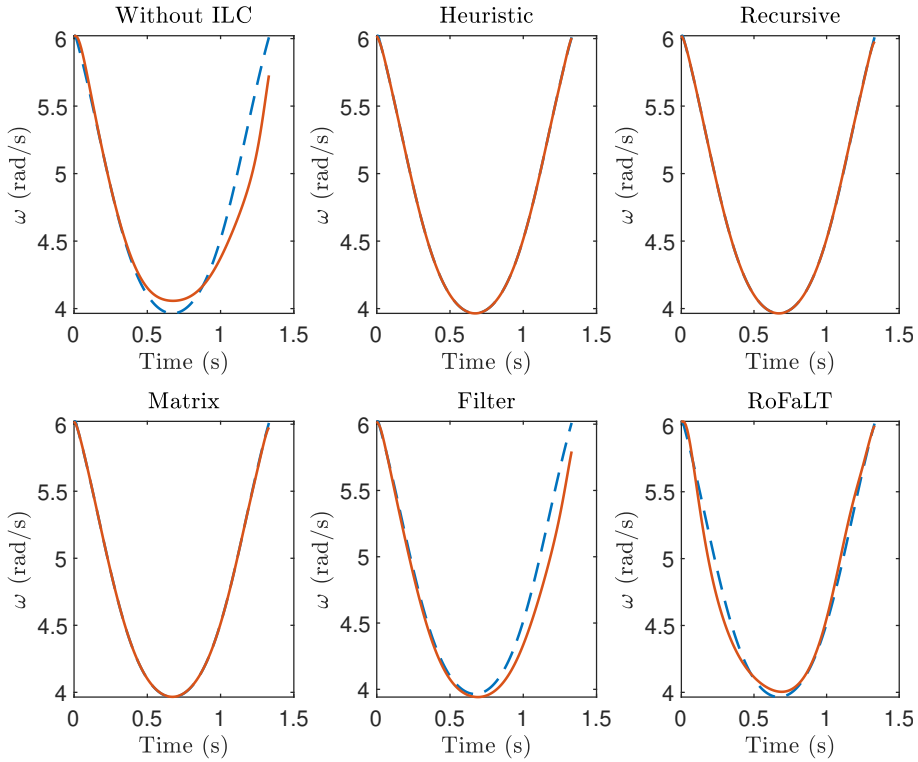
(a) Without splinefit. **(b)** With splinefit.

Figure 4.1: Normalized error L2-norm vs iterations for the different algorithms. The reference is given in Figure 2.9.

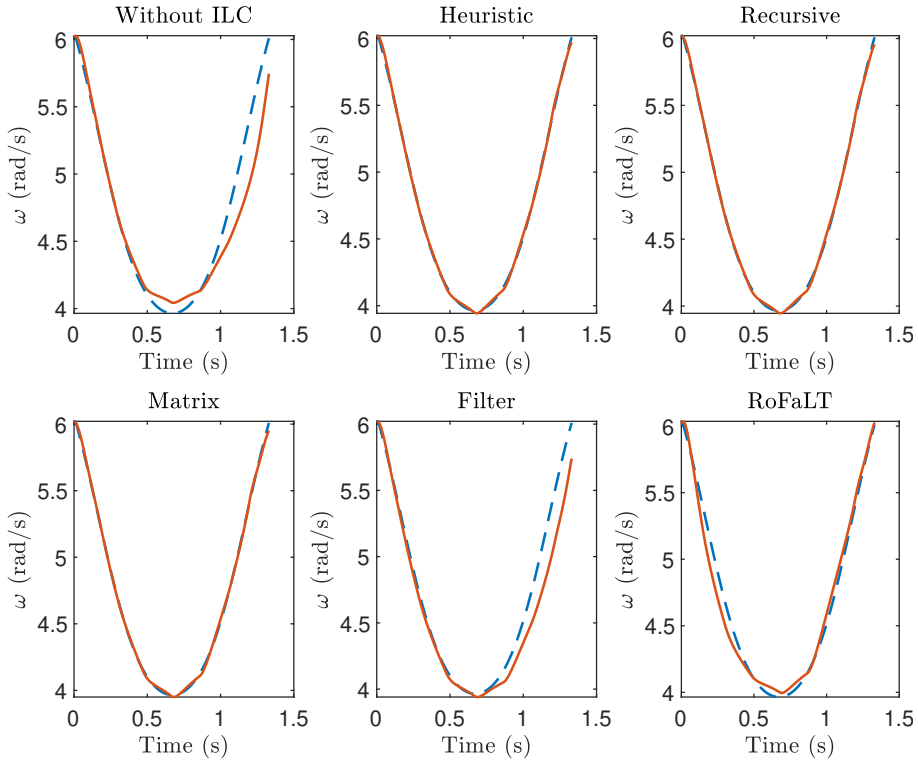
error is larger. With only 8 control inputs, it is not surprising that the algorithm cannot fully compensate for high frequency components. However, the algorithms still converge and the asymptotic error can be judged acceptable for the windshield wiper application. It can also be seen in Figure 4.2b that the reference tracking is good after 10 iterations.

Next we will investigate how the reference signal $\mathbf{r}_k$ develops over iterations. The reference signal after 30 iterations from a simulation using heuristic ILC without splinefit is presented in Figure 4.3a. In Figure 4.3b the reference signal after 30 iterations from a simulation using heuristic ILC with splinefit is presented together with the fitted correction of length 8. When splinefit is not used the reference signal converges after 5 iterations. When splinefit is used, oscillations start to appear in the dense correction signal $\mathbf{u}_k$ in the time domain. However, the fitted correction is still convergent. This is not a desired behavior. Even if the fitted correction seems to be stable in the simulation, problems may occur if the ILC algorithm runs for many iterations.

The splinefit algorithm limits the bandwidth of the ILC controller. Frequency content in the dense correction signal $\mathbf{u}_k$ outside the the bandwidth of the splinefit algorithm will not be included in the fitted reference signal that is applied to the system in the next iteration. Thus, the error in the tracking that caused the frequency content in the correction signal will not be corrected for. Instead, this frequency content will be integrated in $\mathbf{u}_{k+1}$ by the ILC update. Over many iterations, this will cause oscillations in the dense control signal, which can be seen in Figure 4.3b. Notice that, since the frequency of these oscillations is outside the bandwidth of the splinefit algorithm, they will not show in the fitted reference signal or in the tracking. Still, the behavior is undesirable since it can cause numeric issues after many iterations. A solution to the problem is to limit the bandwidth of the ILC algorithm such that it is equal to or lower than the bandwidth of the splinefit algorithm. The heuristic ILC algorithm allows this through the Q-filter. The reference signal after 30 iterations, $\mathbf{r}_{30}$, with the normalized cutoff frequency of the Q-filter

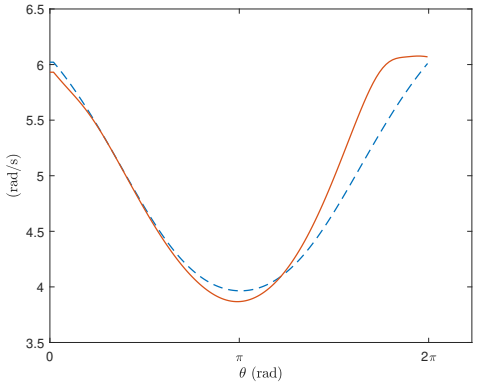


(a) Without splinefit.

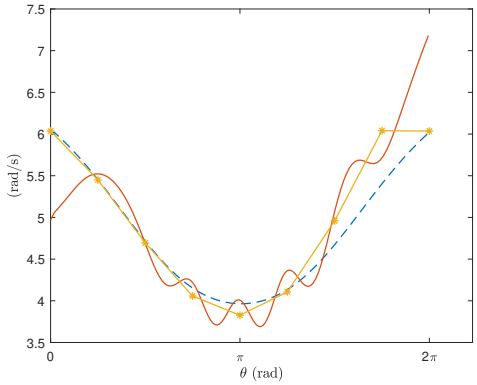


(b) With splinefit.

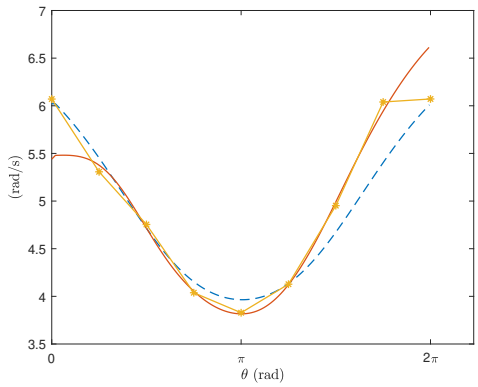
Figure 4.2: The reference tracking (in solid red) after ten iterations of ILC with the different algorithms, as well as the tracking without ILC. The target reference $\mathbf{r}_{0,\mathrm{M1}}$ from Figure 2.9 is shown in dashed blue.



(a) Without splinefit, $\omega_n = 0.2$.



(b) With splinefit, $\omega_n = 0.2$



(c) With splinefit, $\omega_n = 0.05$.

Figure 4.3: The dense reference signal $\mathbf{r}_{30}$ (solid red), the target reference $\mathbf{r}_0$ (dashed blue) and the fitted correction signal (yellow with stars).

chosen as $\omega_n = 0.05$ instead of $\omega_n = 0.2$ is presented in Figure 4.3c. As can be seen, the oscillations are not present when the lower cutoff frequency is used. However, the tracking converges to a slightly larger asymptotic error.

4.3.2 The effect of sampling at constant angle intervals

In Section 3.6 we described why sampling at constant angle intervals instead of with constant sample time can be useful from an implementation perspective. The effect of this modification to the algorithms will now be investigated in simulation. In Figure 4.4, the L2-norm of the tracking error is presented for the investigated algorithms when sampling is performed at constant angle intervals and when the splinefit procedure is used. As can be seen, the asymptotic errors are similar to the results with constant sample time in Figure 4.1b. However, the convergence is slightly slower.

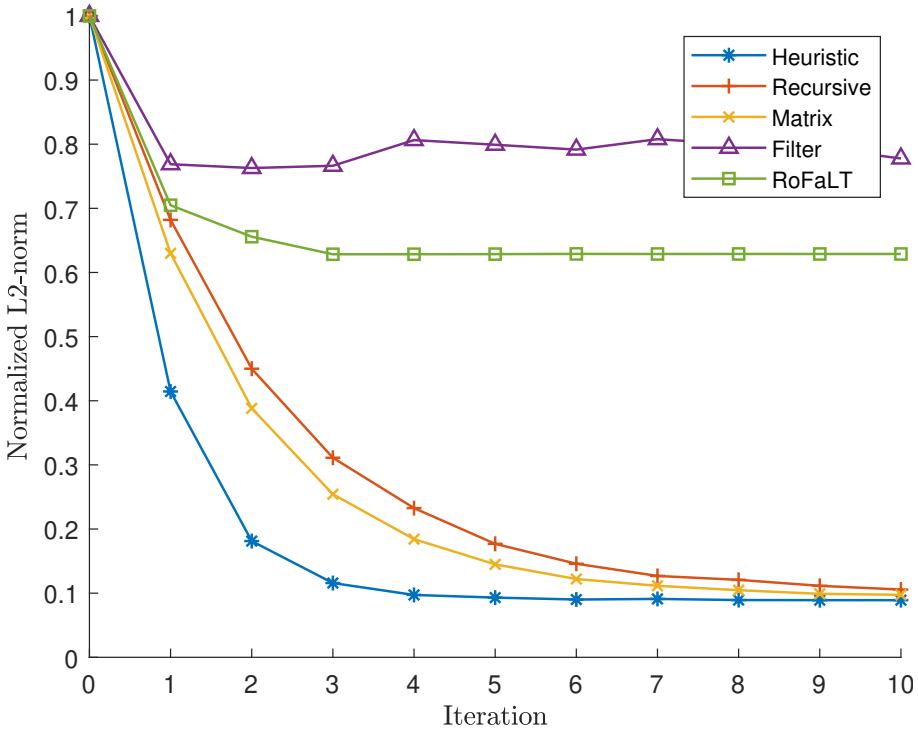


Figure 4.4: Normalized error L2-norm vs iterations for the different algorithms with constant angle sampling and splinefit. The reference is given in Figure 2.9.

4.4 Choice of algorithm

As can be seen in figures 4.1 and 4.4, the heuristic algorithm and the recursive and matrix implementations of norm optimal ILC all perform similarly in the error convergence sense. The heuristic algorithm shows good performance even though it is simple and is thus a good candidate to investigate further in hardware. Another advantage of the heuristic algorithm is that it allows selection of the bandwidth through the choice of Q-filter which makes it suitable to use together with the splinefit procedure. The recursive algorithm also looks promising and is worth investigating further. Due to the large matrices in the matrix implementation, that algorithm is not well suited for embedded applications and will therefore not be included. The filter implementation of norm optimal ILC shows significantly worse performance compared to the matrix and recursive implementation. Therefore, the filter implementation will not be investigated further. The RoFaLT algorithm will also not be investigated further, due to the difficulty of getting good performance, and its implementation difficulty in embedded applications.

5

Experiments

In this chapter we describe some important considerations when implementing the algorithms on a system with a microcontroller. We then present results from experiments on the test rig and compare the results with the results from the simulations in Chapter 4.

5.1 Implementation in hardware

Two algorithms have been implemented on the hardware, heuristic ILC from Section 3.2 and the recursive implementation of norm optimal ILC from Section 3.3.2. Both algorithms were implemented using sampling at constant angle intervals and with the splinefit algorithm from Section 3.5. The chosen ILC algorithms were first implemented in Matlab and then the code generation tool in Matlab was used to generate C-code. The C-code was then integrated into the existing control system software.

The test rig used for the experiments is shown in Figure 5.1. The control unit has a single core Infineon XMC4500 Arm processor with clock frequency 120 MHz and 160 KiB of RAM. The processor and the memory must be shared with the existing control system. Approximately 90 KiB of RAM is available to the ILC algorithm.

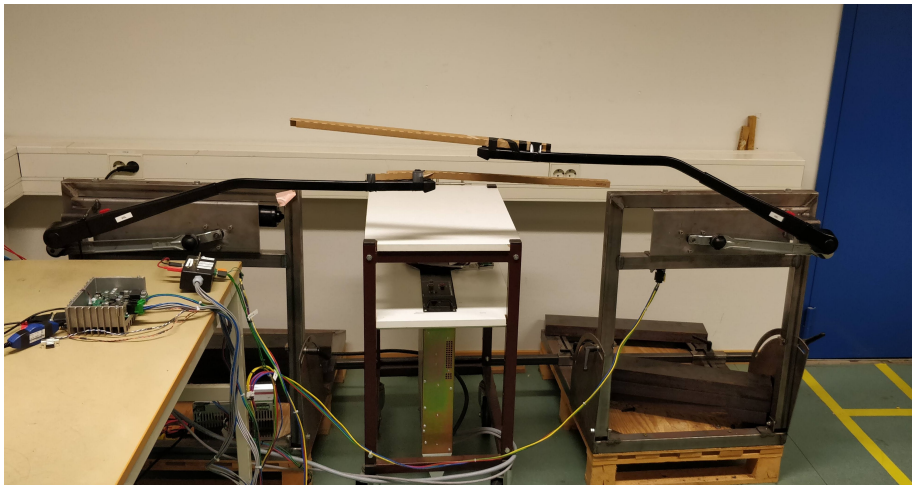


Figure 5.1: A picture of the test rig.

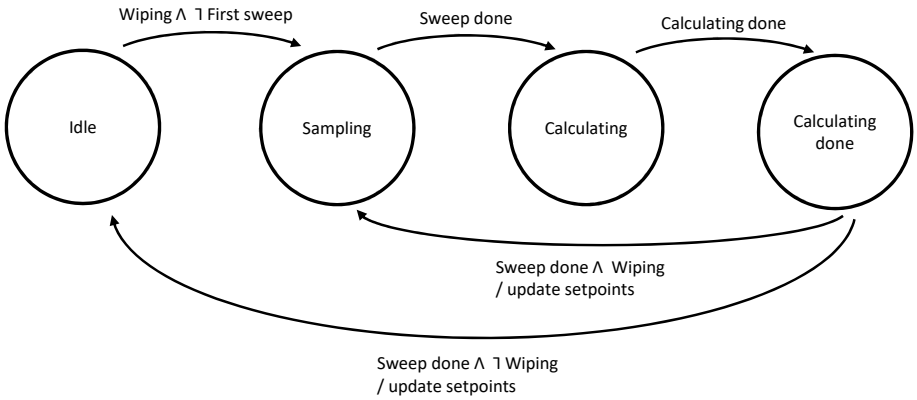


Figure 5.2: A state machine describing the sampling and calculation process in the microcontroller. The / denotes taking an action.

From tests it was discovered that the heuristic ILC update algorithm has a computation time of 8.3 ms on the hardware while the recursive implementation of norm optimal ILC has a computation time of 8.5 ms. Of this, the splinefit procedure accounts for 6.9 ms. This is the execution time for each motor, thus the total execution time is approximately 16 ms. This includes interrupts from higher priority tasks. When implemented with 32-bit floating point variables, the heuristic algorithm requires 27 KiB of memory for two motors and the recursive algorithm requires 30 KiB of memory for two motors. If a first order model is used instead of second order model the required memory for the recursive algorithm is reduced to 26 KiB and the computation time is reduced to 7.7 ms. The reported memory requirements include both RAM and flash memory where about two thirds are RAM, which is well within the available capacity.

The system runs continuously and we do not want to update the reference setpoints after a sweep has started. Further, 16 ms is too much for the algorithm to be executed in the short time period between wiper sweeps. Therefore, the ILC update is performed every other wiper sweep. Measurements are sampled during an entire sweep and then the calculations are performed in the background during the next sweep. The reference setpoints are then updated at the end of the calculation sweep. To demonstrate this behavior, a state machine is presented in Figure 5.2.

In ILC it is assumed that the initial state is invariant over iterations. In simulation this was achieved by initializing the speed of both wipers to a reference value at the start of each iteration. In practice, the initial speed of the wipers will depend on the speed at the end of the previous sweep. This can still fulfill the relaxed assumption in Section 3.1.2, that the initial error is bounded by some $\epsilon > 0$. The exception is the first sweep where the initial state is zero. This will cause a large error in the ILC update. To avoid this, the ILC algorithm is not enabled until the second wiper sweep.

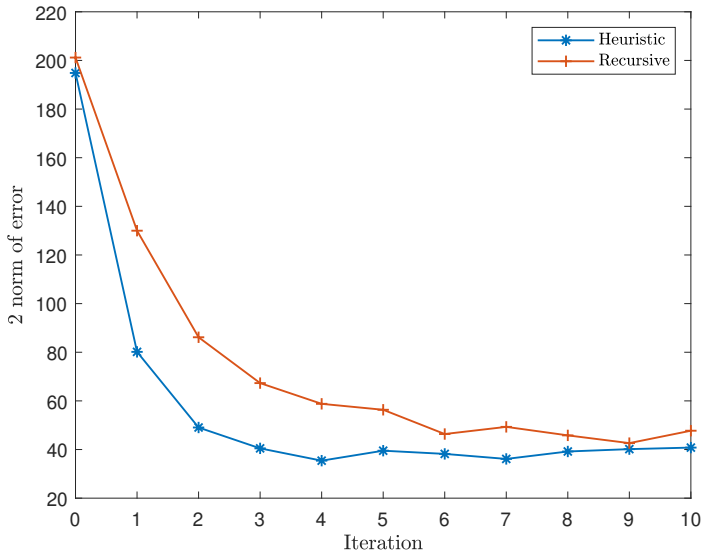


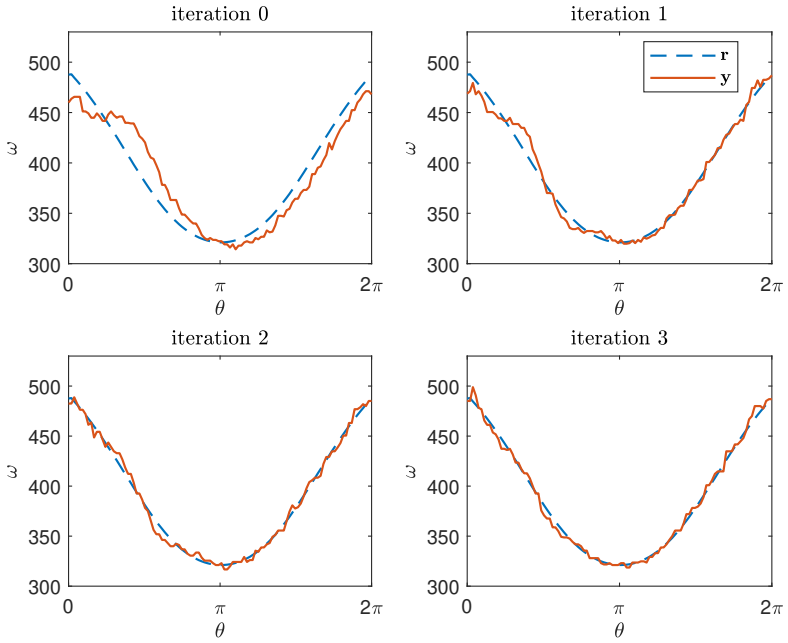
Figure 5.3: The L2-norm of the tracking error over 10 iterations using the heuristic and the recursive algorithm.

5.2 Experimental results

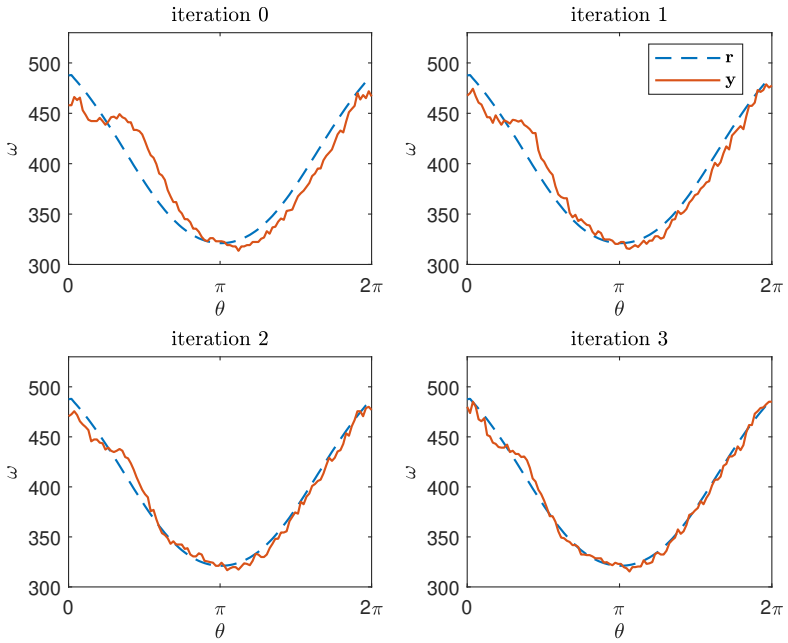
In Figure 5.3 the L2-norm of the tracking error from two experiments in the test rig are presented. The plot shows results from one experiment with heuristic ILC and one experiment with the recursive implementation of norm optimal ILC. The L2-norms are not normalized since the initial error is slightly different in the two experiments. As in the simulations, results are presented for one motor only and the results for the second motor are analogous. The convergence properties are similar to those from the simulation. In Figure 5.4 the reference tracking over three ILC iterations is presented for the two algorithms. It can be seen in the figures that the convergence is slightly faster for the heuristic algorithm but that the asymptotic error is similar. This is consistent with the simulation results in Chapter 4. In Figure 5.4 one can see that the ILC algorithms compensate for a phase shift from the PI controller as well as disturbances in the tracking from the nonlinear dynamics.

In Figure 5.5 measurements of gearbox angles for the two motors are presented. The measurements are plotted together with the sigmoid reference from Section 2.2.1 and the areas where the wipers clash. As can be seen, the wipers are almost clashing during the first sweep. After three ILC iterations the tracking of the sigmoid curve has improved and the wipers do not clash. This experiment shows that good tracking of the reference curve defined in the θ_1-θ_2-domain is achieved when ILC is applied separately to the two motor velocities.

The sweep frequency after 10 ILC iterations was measured as 43 Hz for both algorithms, which is slightly lower than the 45 Hz that the reference curve was designed for.

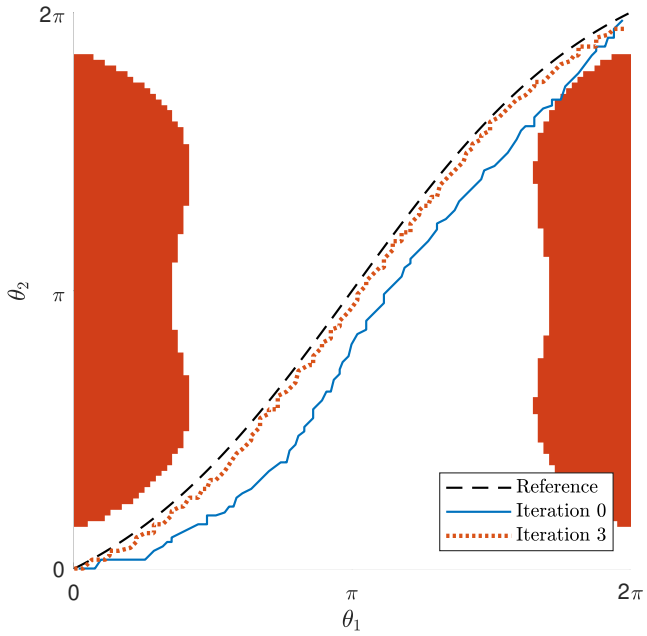


(a) Heuristic algorithm.

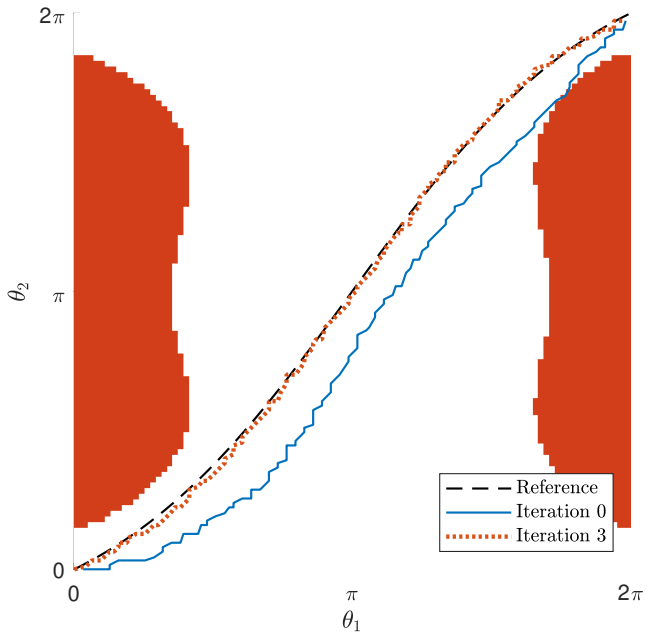


(b) Recursive algorithm.

Figure 5.4: Velocity reference tracking from an experiment in the test rig. Plots are shown for iteration zero (before ILC) to iteration three.



(a) Heuristic algorithm.



(b) Recursive algorithm.

Figure 5.5: Measurements of gearbox angles from two experiments in the test rig for iteration 0 (before ILC) and iteration 3, together with the areas where the wipers clash (red).

6

Discussion

In this chapter we discuss the results found in chapters 4 and 5 and their implications on a system level.

6.1 The splinefit algorithm

The splinefit algorithm presented in Section 3.5 makes it possible to apply ILC directly to the eight setpoints that are used for manual tuning. This is practical since the tuning then easily can be copied to another unit and only small modifications are required to the present control system. In Chapter 4 we showed that the splinefit algorithm reduces the performance of the algorithm both in a sense of stability and the size of the asymptotic tracking error. Moreover, in Section 5.1 we showed that the splinefit algorithm adds significantly to the execution time of the ILC update algorithms. Thus, by not using the splinefit algorithm and instead applying the dense control signal directly to the system one would save computation time, reduce the memory requirements and potentially improve the performance of the system. However, this requires some modifications to the present control system.

To mitigate the computational downsides of the splinefit algorithm, an alternative approach could be to use a simpler downsampling scheme, such as a decimation procedure. The increased loss of information in the correction signal could be compensated for by the learning algorithm's iterative nature, resulting in similar performance. However, this type of scheme cannot guarantee that the downsampled correction signal is periodic.

6.2 System use cases

When the problem with individual tuning of the wiper system was introduced in Section 1.1.1, three types variations in the system dynamics were described: variations between different bus models, variations between individual units of the same model and variations over time for one unit. Which of these variations the ILC algorithm learns to compensate for depends on how the algorithm is used. Based on this, we see three possible use cases for the ILC algorithm.

In the first use case the algorithm is used once for each new bus model, replacing the manual tuning process used today. Then the algorithm learns to compensate

for variations between different bus models. Here, the ILC algorithm makes the development process easier, allowing a system agnostic development process.

In the second use case, the algorithm is used once for each bus in the factory. This way, the algorithm can compensate for variations between individual units of the same model. A drawback of this use case is that the algorithm becomes a part the manufacturing process, which seems impractical from a production perspective.

In the third use case, the algorithm runs continuously during all operation to compensate for time varying dynamics. This puts stricter requirements on the tuning, putting an emphasis on long term stability. To achieve stability over many iterations the algorithm should be tuned for robustness through e.g. the Q-filter. If the splinefit algorithm is used, the cutoff frequency of the Q-filter should be reduced further to achieve stability, as shown in Section 4.3.1. However this will increase the asymptotic error of the tracking.

In the first and second use cases, the algorithm is used once and then turned off after a few iterations, making stability over hundreds of iterations unnecessary. Then, the algorithm can be tuned for fast convergence and small asymptotic error.

6.3 Choice of velocity references

In Section 2.2.1 a systematic approach to choosing velocity references which ensure that the wipers do not clash and that a specified cycle time is achieved was presented. In Chapter 5 the cycle time achieved after ILC was slightly longer than specified due to imperfect tracking. To achieve the specified cycle time, a slightly shorter cycle time can be chosen in the design process.

A drawback to the method presented in Section 2.2.1, compared to the manual tuning process that is used today, is that it does not capture some of the more subjective objectives of the tuning process. When an experienced engineer tunes the wiper system, objectives such as how esthetic and symmetric the wiping appears are considered. These objectives are inherently difficult to formulate mathematically. A possible alternative is to measure the output from a well-behaved and good looking hand tuned wiper system and then use this as the reference that the ILC algorithm should learn. This way, the system can learn to follow a reference that fulfills these objectives. The ILC algorithm enables the system to learn to follow this trajectory even if the dynamics of the system are unknown which removes the need for individual tuning for each new bus model.

7

Conclusion

The results from simulations in Chapter 4 and experiments in the test rig in Chapter 5 show that iterative learning control can be used to learn a set of reference setpoints that improves the tracking of a velocity reference. With iterative learning control, good tracking can be achieved even when the dynamics of the system is not fully known. This simplifies the tuning process for new bus models because the wiper system can automatically adapt to and compensate for the unknown load characteristics of a new bus model.

The tuning process for new bus models then concerns the choice of the reference curves that the ILC algorithm should learn to follow. In Section 2.2.1 we presented a systematic approach to choosing these references in a way that guarantees that the wipers do not clash and that a specified cycle time is achieved. We believe that this design space is more intuitive and easier to work with than separate speed references.

The method we propose in this work is best suited as a development tool for a more structured way of working with the tuning. Although it can be generalized to continuous use as discussed in Chapter 6, we believe that the most promising use case is the tuning tool.

7.1 Future work

In this thesis we have only worked with a single hardware platform. To fully verify the usefulness of the proposed method, further research is needed. Specifically, the results obtained from the proposed method needs to be verified in end use scenarios, such as on a windshield under different weather conditions and during driving scenarios.

Bibliography

[1] M. Anderson, "Window-cleaning device.", US743801A, 1903.

[2] The European Union, "COMMISSION REGULATION (EU) No 1008/2010", *Official Journal of the European Union*, 2010. [Online]. Available: https://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=OJ:L:2010:292:0002:0020:EN:PDF.

[3] K. L. Richards, "34. Introduction to Linkages", in *Design Engineer's Sourcebook*, CRC Press, 2018, ch. 34, ISBN: 978-1-4987-6341-7. [Online]. Available: https://app.knovel.com/hotlink/pdf/id:kt011MJYE3/design-engineers-sourcebook/introduction-linkages.

[4] ——, "36.11 Referred Moment of Inertia (IReferred)", in *Design Engineer's Sourcebook*, CRC Press, 2018, ch. 36, ISBN: 978-1-4987-6341-7. [Online]. Available: https://app.knovel.com/hotlink/khtml/id:kt011MK0O4/design-engineers-sourcebook/referred-moment-inertia.

[5] D. Eberly. (Apr. 2, 2019). Moving Along a Curve with Specified Speed, [Online]. Available: https://www.geometrictools.com/Documentation/MovingAlongCurveSpecifiedSpeed.pdf (visited on Feb. 14, 2020).

[6] D. A. Bristow, M. Tharayil, and A. G. Alleyne, "A survey of iterative learning control", *IEEE Control Systems Magazine*, vol. 26, no. 3, pp. 96–114, 2006, ISSN: 1941-000X. DOI: 10.1109/MCS.2006.1636313.

[7] M. Norrlöf, "Iterative Learning Control: Analysis , Design , and Experiments", Ph.D dissertation, Linköpings universitet, 2000, ISBN: 9172198370. [Online]. Available: http://www.control.isy.liu.se.

[8] M. Norrlof, "An adaptive iterative learning control algorithm with experiments on an industrial robot", *IEEE Transactions on Robotics and Automation*, vol. 18, no. 2, pp. 245–251, 2002, ISSN: 2374-958X VO - 18. DOI: 10.1109/TRA.2002.999653.

[9] S. S. Saab, "A stochastic iterative learning control algorithm with application to an induction motor", *International Journal of Control*, vol. 77, no. 2, pp. 144–163, Jan. 2004, ISSN: 0020-7179. DOI: 10.1080/00207170310001646282. [Online]. Available: https://doi.org/10.1080/00207170310001646282.

[10] A. D. Barton, P. L. Lewin, and D. J. Brown, "Practical implementation of a real-time iterative learning position controller", *International Journal of Control*, vol. 73, no. 10, pp. 992–999, Jan. 2000, ISSN: 0020-7179. DOI: 10.1080/002071700405941. [Online]. Available: https://doi.org/10.1080/002071700405941.

[11] S. Arimoto, "A Brief History of Iterative Learning Control", in *Iterative Learning Control: Analysis, Design, Integration and Applications*, Z. Bien and J.-X. Xu, Eds. Boston, MA: Springer US, 1998, pp. 3–7, ISBN: 978-1-4615-5629-9. DOI: 10.1007/978-1-4615-5629-9_1. [Online]. Available: https://doi.org/10.1007/978-1-4615-5629-9%7B%5C_%7D1.

[12] J. Wallén, "Estimation-based iterative learning control", PhD thesis, Linköping University, Linköping, 2011, ISBN: 9789173932554. [Online]. Available: www.control.isy.liu.se.

[13] N. Amann, D. H. Owens, and E. Rogers, "Iterative learning control for discrete-time systems with exponential rate of convergence", *IEE Proceedings - Control Theory and Applications*, vol. 143, no. 2, pp. 217–224, 1996, ISSN: 1350-2379. DOI: 10.1049/ip-cta:19960244.

[14] S. Gunnarsson and M. Norrlöf, "On the design of ILC algorithms using optimization", *Automatica*, vol. 37, no. 12, pp. 2011–2016, Dec. 2001, ISSN: 00051098. DOI: 10.1016/S0005-1098(01)00154-6.

[15] D. H. Owens, *Iterative Learning Control*, ser. Advances in Industrial Control. London: Springer London, 2016, ISBN: 978-1-4471-6770-9. DOI: 10.1007/978-1-4471-6772-3. [Online]. Available: http://link.springer.com/10.1007/978-1-4471-6772-3.

[16] A. Steinhauser, T. D. Son, E. Hostens, and J. Swevers, "RoFaLT: An optimization-based learning control tool for nonlinear systems", in *Proceedings - 2018 IEEE 15th International Workshop on Advanced Motion Control, AMC 2018*, Institute of Electrical and Electronics Engineers Inc., Jun. 2018, pp. 198–203, ISBN: 9781538619469. DOI: 10.1109/AMC.2019.8371087.

[17] N. Golovchenko. (2004). Least-squares Fit of a Continuous Piecewise Linear Function, [Online]. Available: https://golovchenko.org/docs/ContinuousPiecewiseLinearFit.pdf (visited on Feb. 10, 2020).

[18] J. Lundgren. (2020). SPLINEFIT, [Online]. Available: https://se.mathworks.com/matlabcentral/fileexchange/71225-splinefit (visited on Apr. 28, 2020).

A

Model parameters

Table A.1 contains the parameters used for systems modeling in Chapter 2.

Table A.1: Model parameters used for the system

Parameter	Symbol	Value
Wiper mass	J_θ	5 kg
Inertia rotating side	J_θ	0.001 kgm^2
Inertia reciprocal moving side	J_φ	5 kgm^2
Wiper friction	b_{wiper}	22 N $\cdot$ (rad/s)$^{-1}$
Motor friction	b_{motor}	$1.5 \cdot 10^{-5}$ N $\cdot$ (rad/s)$^{-1}$
Motor resistance	R_m	0.8 Ω
Motor inductance	L_m	0.0015 H
Torque constant	ψ	0.0357 Wb
Gear ratio	N_{GB}	81
Proportional gain	K_p	0.7325 Vs
Integral gain	K_i	30.5 Vs
Wiper point mass offset	r	1 m
Axes offset motor 1	$\ell_{0,M1}$	0.309 m
Crank length motor 1	$\ell_{1,M1}$	0.066 m
Connector length motor 1	$\ell_{2,M1}$	0.320 m
Rocker length motor 1	$\ell_{3,M1}$	0.100 m
Axes offset motor 2	$\ell_{0,M2}$	0.336 m
Crank length motor 2	$\ell_{1,M2}$	0.066 m
Connector length motor 2	$\ell_{2,M2}$	0.345 m
Rocker length motor 2	$\ell_{3,M2}$	0.098 m
Wiper arm length motor 1	$L_{1,M1}$	0.47 m
Wiper blade length motor 1	$L_{2,M1}$	0.86 m
Wiper arm to wiper blade angle motor 1	α_{M1}	149°
Wiper arm length motor 2	$L_{1,M1}$	0.46 m
Wiper blade length motor 2	$L_{2,M2}$	0.85 m
Wiper arm to wiper blade angle motor 2	α_{M2}	160°