Embedded Systems Project

Adaptive Racer

Linus Wågberg, Lars-Gunnar Olofsson, Erik Forslid, Mikael Sahlin

January 22, 2014

Abstract
In this paper ways of improving the performance of a robot racer around a set closed track was tested. First by letting the robot learn from previous laps to improve its tracking by iterative learning control (ILC). Secondly by removing the slip of the wheels to improve the stability of the robot by traction control. The tests with the ILC resulted in a 57% reduction of the RMS error. The traction control resulted in a reduction of virtually all measurable slip for high wheel speeds.
Contents

List of Figures 3

1 Introduction 4

2 Theory 5
  2.1 PID-regulator ................................................. 5
  2.2 ILC-regulator ................................................ 5
    2.2.1 ILC-design .............................................. 5
    2.2.2 PD-Type and Tunable Designs .......................... 6
  2.3 Traction control ............................................. 7

3 Method 9
  3.1 Programming Environment ..................................... 9
    3.1.1 Integrated Development Environment (IDE) ................. 9
  3.2 Modeling of the system ....................................... 9
    3.2.1 Mechanical modeling of the racer .......................... 10
    3.2.2 Identifying the system parameters ........................ 12
    3.2.3 Track .................................................... 13
  3.3 ILC ............................................................ 14
  3.4 Traction Control ............................................. 15
    3.4.1 Introducing slip ......................................... 16
    3.4.2 Calibration of slip and slip threshold ...................... 17

4 Result 18
  4.1 ILC implementation .......................................... 18
  4.2 Traction control ............................................ 19
  4.3 ILC and Traction Control ................................... 20

5 Discussion 22
  5.1 ILC-Algorithm ............................................... 22
  5.2 Traction control ............................................ 22
  5.3 Programming issues .......................................... 23

6 Conclusion 24

7 Work distribution 24

8 References 25
List of Figures

1. System model ................................................. 6
2. Showing non monotonically converging error as a consequence of an ILC algorithm .................................... 7
4. The communication scheme between the NXT brick and the computer ......................................................... 9
5. System model ..................................................... 10
6. Distance from the center of the track .......................................................... 10
7. Final system model of the racer ...................................................... 12
8. Distance from axle to sensor ........................................... 12
9. Output when $\Delta u$ is an unit step ............................................. 13
10. Track with gradient and dashed lines. ........................................... 13
11. The intensity on the gradient from our track ......................... 14
12. ILC system ................................................. 15
13. Traction control system ............................................... 16
14. To introduce an difference in slip, the same tyres was used with and without rubber. The maximum voltage was set to 40% .................................................. 17
15. The RMS error for different proportional gains in the first order proportional ILC, the values are splined to improve quality ........................................ 18
16. The results with the best improvement using a first order proportional ILC ........................................... 19
17. Calculated slip with an maximum voltage of 50 and 70 percent see figure 17 a and b. Traction control is enabled after 10 laps and the signal is low-passed filtered in Matlab to highlight the trends ........................................ 19
18. Estimated wheel and chassis speed with an maximum voltage of 50 and 70 percent see figure 18 a and b. Traction control is enabled after 10 laps and the signal is low-passed filtered in Matlab to highlight the trends ........................................ 20
19. Root mean square error with and without traction control enabled. With the traction control off, the vehicle starts to have problem to stay on the track ........................................ 20
20. Root mean square error with and without traction control enabled. With the traction control off, the vehicle starts to have problem to stay on the track ........................................ 21
1 Introduction

Adaptive control is a wide set of methods to regulate a system that changes over time. In the examined case, the road conditions for a line following robot will change. Different surfaces will be applied to different parts of the track, and the task is to examine and compare two different methods. Firstly, using iterative learning control (ILC) to compensate for the reoccurring errors, and secondly by applying traction control to the system in order to regulate the speed in order to maintain control of the vehicle. ILC has successfully been used in industrial applications for many years. By learning from previous attempts, it minimizes the errors that can occur over time. This kind of technique is commonly seen in robotic arms that perform the same set of motions over and over again.

Traction control is a widely researched topic in the automobile industry and comprises ABS breaks, anti-skid systems, etc. Because of its importance in safety and performance, traction control theory places a lot of emphasis on this research. Measurements and estimations are the main problems regarding chassis speed and friction coefficient. There are several different methods depending on fuel source and the number of non-controlled wheels. For electrical vehicles, it is common to measure the torque since this correlates well with input current, which is easily measured. The speed estimation problem is commonly solved by measuring the rotation of the non-controlled wheels.

In this paper, a line following robot is put to the test by altering the road conditions and comparing the performance of a simple traction control and an ILC algorithm. The aim being to improve the performance of the robot's ability of following a set track.
2 Theory

2.1 PID-regulator

The PID-controller is a widely used feedback controller, especially in industrial control systems. PID stands for proportional-integral-derivative, every part have different parameters that have to be calculated for the controller.

- **P**: Dependent on the present error
- **I**: Sums up the past errors
- **D**: Predicts the upcoming errors by looking at the current pace of change

The general appearance of the PID-controller is

\[ u(t) = K_p e(t) + K_i \int_0^t e(\tau) d\tau + K_d \frac{d}{dt} e(t) \]

In the equation you find the constants for each part of the controller, \( K_p \), \( K_i \) and \( K_d \). The output of the system is \( u(t) \).

2.2 ILC-regulator

The ILC-controller differs from other learning types such as neural networks, adaptive control and repetitive-control. Both Neural networks and the adaptive controller modifies the control parameters rather than the output. In addition the adaptive controller typically do not take advantage of the previous iteration and Neural networks need large quantities of data where convergence could be difficult. RC and ILC is quite similar with an difference of setting initial condition on each trial, ILC sets the same value every cycle and RC sets the final condition in the end of the cycle. This leads to different types of results and analyse techniques. As mentioned earlier ILC-controllers has been proven very successfully applied in industry robots performing an repetitive task, which suits our needs perfectly.

2.2.1 ILC-design

There is four different types of ILC-controller designs:

- **PD-Type and Tunable Designs**, is an design that can be applied without extensive knowledge of the system.
- **Plant Inversion Methods**, relies heavily on the modelling of the system and can be sensitive to model errors.
- **\( H_{\infty} \) Methods**, can be used to construct an robust and reliable controller with the expense of the performance.
- **Quadratically Optimal Design**, uses an quadratic performance criterion to get the optimal controller.
When choosing design the PD-Type and Tunable controller was the best fit, basically because it can handle small errors of the system. Our robot uses number of light sensors and they haven’t been so accurate when the light conditions change over day in which has lead to difficulty mapping the track.

2.2.2 PD-Type and Tunable Designs

The name implies that it is an PD function containing an proportional and derivative part. The integrating part is missing because it is an learning function and it has a natural integration from the first cycle to the next. The time discrete function is written like:

\[ u_{j+1}(t) = u_j(k) + k_p e_j(k + 1) + k_d [e_j(k + 1) - e_j] \]

Ideally the controller needs to converge over a couple of cycles and an low-pass filter \( Q \) see figure 1 can disable high frequency’s helping the controller to meet the condition. The low-pass filter will also contribute to robustness as well as high frequency noise filtering. The cut of frequency has to be tuned in relation to the system.

![Figure 1: System model](image)

The convergence time will depend on how many cycles it will take before the ILC-controller starts to minimize the error. It is important to configure a system that can handle an bigger error see figure 2, because it will take a few iterations depending on the system before the ILC-controller has detected all the uncertainties and started to comprehend them hence improving the error [4].
2.3 Traction control

When talking about traction control it is usually referred to the lateral movement where different traction control systems such as ABS and anti-spin systems operates. The longitudinal movements mostly rely on the driver and the car dynamics and hence are not discussed further in this paper.

Traction control is a broad subject which is researched widely within the automobile industry. There are several different approaches to this problem, in general a comparison between the rotational speed of the tires and that of the chassis of the vehicle is made. The difference in these is a measurement of slip and is calculated as follows,

\[ \lambda = \frac{r\omega - v_{\text{chassis}}}{r\omega} \]  

where this equation is named the "Magic formula" and is commonly referred to in the literature on the topic. Many of the approaches to the traction control problem focuses on estimating the friction from the torque. Then a relationship between the slip and friction coefficient is examined. Studying figure 3 its easy to see that in order to maintain grip we need the friction-slip curve to have a positive derivative if exceeded the torque needs to be lowered.

Controlling torque works very well for electrical vehicles since the output torque and the input voltage are well correlated. With combustion vehicles the input power to the engine depends on a lot of other factors than just the amount of fuel combusted so torque estimation is not as straight forward but still used in many systems.

The biggest problem one faces when trying to solve the traction problem is determining the friction. Almost all methods work with estimating the friction coefficient through the torque which in itself can be troublesome. Depending on how the vehicle is driven speed might not be measurable and then this have to be estimated as well.
Sensor based friction coefficient estimation:
There has been studies and experiments with different sensors in front of the wheels in order to predetermine the upcoming road conditions.

1. **Optical and acoustical sensors** have been used to some success and have a promising feature in detecting wet areas on the surface of the road and thereby help prevent aqua-planning.

2. **Treads** have also been used on the wheels but affordable solutions have not been found. These have also proven technically very complex and hence have not been used in any bigger extent.

Speed Estimation:
As mentioned above the common way to measure the speed of the vehicle is measuring the rotational speed of the non-driven wheel. Where this is not possible sensor fusion algorithms are used. For instance merging ABS, gyro, accelerometer and engine data is discussed in [6]. Here they use kalman filters with change detection by the above mentioned sensors.

Controller methods, drawbacks and benefits

1. **MFC**: Uses few sensors which leads to high reliability and higher independence from driving conditions. This however is a compensation-based method and have to consider the worst stability case to decide on the compensation gain. Tuning the gain for tire-road conditions is also problematic and is a big drawback for the practicality of this method.

2. **MTTE**: Uses only torque and wheel rotation as inputs and from this information estimates the maximum allowed torque to maintain grip, hence the Maximum Transmissible Torque Estimate. Relies on a good torque estimation.
3 Method

3.1 Programming Environment

When using an NXT brick there are several options to program it. The following were considered in this project:

- **Robot C** is a simplified version of C where it uses the Robot C environment and Application Programming Interface (API) to program the NXT brick.

- **LejOS** is a system based on the Java Runtime Engine. This uses a pre-written API of classes in the Java programming language to program the NXT.

- **LabView** is a graphical programming language which Lego uses as their main consumer IDE.

When reading previous encounters with the NXT brick as well as taking comments from other colleges into account the decision to go with LejOS was made. Java is also the programming language which all of the members of the group felt most comfortable with.

3.1.1 Integrated Development Environment (IDE)

Eclipse is an IDE which has many useful integrated features, one being a "store" in which more extensions can be downloaded. LejOS is one of those extensions. In the beginning Eclipse was downloaded as well as extensions to work with LejOS. Further more the NXT brick was flashed to install the Java environment on the brick, this is a requirement when using LejOS. The brick then was paired via bluetooth to enable Eclipse to send code to the brick wirelessly and for the brick to start sending a real time logging back to the computer, this is shown in Figure 4.

![Figure 4: The communication scheme between the NXT brick and the computer](image)

3.2 Modeling of the system

The signal we are able to control is the voltage input into the motors, which is expressed as a percentage of the maximum power in the control program. Building the model with two motors we get an input into the right engine \( u_r \) and an input into the left engine \( u_l \).

So the translational speed depends on the input into both engines \( u_0 \) and the rotational depends on the difference between the inputs

\[
\Delta u = u_l - u_r
\]
From this we get the following inputs in our system

\[ u_l = u_0 + \frac{\Delta u}{2} \]

\[ u_r = u_0 - \frac{\Delta u}{2} \]

Furthermore we are interested in following a line on our track and for this we use LEGO’s light sensor and denote this our output signal \( y \).

3.2.1 Mechanical modeling of the racer

Two sets of motion is identified when modeling the movement of the vehicle. A translational motion and a rotational motion. The output of the system, \( y = y_{lat} + y_{rot} \), corresponds to the distance to the center of the track, where \( y_{lat} \) is the lateral displacement and \( y_{rot} \) is the angle from center of the track.

\[ y_{1}(t) = \int_{0}^{t} v \cdot \sin(\phi(\tau)) d\tau \]

Figure 5: System model

Figure 6: Distance from the center of the track
\[ y_2(t) = L \cdot \sin(\phi(\tau)) \]

Where an approximation for small angles is made, \( \sin(\phi) \approx \phi \) is made. And \( y(t) \) can be written as

\[ y(t) \approx L \cdot \sin(t) + v \int_0^t \phi(t) dt, \quad v = Ka u_0. \] (2)

Here \( K_u \) is a constant depending on the fixed value \( u_0 \). A Laplace transform of 2 leads us to the transfer function shown in equation 3

\[ Y(s) = G_R(s) \Phi(s) \]

\[ G_R(s) = L + \frac{K_u u_0}{s}. \] (3)

Looking at the rotational motion and using the laws of mechanics we can express the rotation of the vehicle \( \Phi \) as a function of the input voltage \( U \) into the engines. The following notations will be used in order to explain these relationships

<table>
<thead>
<tr>
<th>T</th>
<th>torque</th>
</tr>
</thead>
<tbody>
<tr>
<td>( f_{r,l} )</td>
<td>friction on right/left wheel</td>
</tr>
<tr>
<td>( \theta_{r,l} )</td>
<td>angle of right/left wheel</td>
</tr>
<tr>
<td>( \tau_{r,l} )</td>
<td>torque on right/left wheel</td>
</tr>
<tr>
<td>( r )</td>
<td>radius of wheel</td>
</tr>
<tr>
<td>( 2R )</td>
<td>distance between wheels</td>
</tr>
<tr>
<td>( \tau_F )</td>
<td>torque to compensate for friction and back emf of the motor, proportional to ( \tau_F = K_b \dot{\theta} )</td>
</tr>
<tr>
<td>( \tau_\omega )</td>
<td>torque to produce angular acceleration of wheel, proportional to ( \tau_\omega = I_\omega \ddot{\theta} )</td>
</tr>
<tr>
<td>( (\tau_{\text{tot}})_{r,l} )</td>
<td>total torque that the right/left motor will exert, proportional to ( (\tau_{\text{tot}})<em>{r,l} = K_r u</em>{r,l} )</td>
</tr>
<tr>
<td>( \tau_L )</td>
<td>torque to compensate the load of the vehicle, explanation given below</td>
</tr>
</tbody>
</table>

The mechanical laws used is listed below

\[ T = I \ddot{\phi} \]
\[ T = (f_l - f_r) R \]
\[ f_r = \frac{\tau_r}{r} \]
\[ f_l = \frac{\tau_l}{r} \]
\[ \phi = r \frac{(\theta_l - \theta_r)}{2R}. \] (4)

These combined gives the following relationship

\[ (\tau_r - \tau_l) \frac{R}{r} = I \ddot{\phi}. \] (5)

The relationship between the torques on each wheel is given by

\[ \tau_{\text{tot}} = \tau_\omega + \tau_F + \tau_L \]
\[ \tau_L = K_r - I \ddot{\theta} - K_b \dot{\theta} \] (6)
Equation 6 plugged into 5 and using the last equation in 4 can be written as

\[
\left(\frac{rI}{R} + \frac{2Rr}{r}I_\omega\right)\ddot{\phi} = -\frac{2R}{r}K_b\dot{\phi} + K_r\Delta u
\]  \hspace{1cm} (7)

where \(\Delta u = u_i - u_r\). \(u(t) = K_\Delta \Delta u(t)\) where \(K_\Delta\) is a constant. This dynamic second order equation which Laplace transform can be written as

\[
\Phi(s) = G_T(s)U(s) = \frac{K}{s(Ts + 1)}U(s)
\]  \hspace{1cm} (8)

where

\[
K = \frac{K_\Delta K_r}{2R K_s}
\]  \hspace{1cm} (9)

\[
T = \frac{1}{K_s} \left(\frac{r^2 I}{2R^2} + I_\omega\right)
\]

The final transfer function between \(U(s)\) and \(Y(s)\) can be expressed as a series of \(G_R\) and \(G_T\) as seen in figure 7.

\[
G(s) = G_R(s)G_T(s) = (L + \frac{K_u}{s}u_0)\left(\frac{K}{s(Ts + 1)}\right) = K\frac{Ls + v}{s^2(Ts + 1)}
\]  \hspace{1cm} (10)

### 3.2.2 Identifying the system parameters

The final transfer function of the system (equation 9) contains unknown parameters \(L\) (Length), \(v\) (Velocity), \(K\) (constant) and \(T\) (Time). These parameters can be identified by different types of physical measurements of the robot.

- Length \(L\) can be measured by the distance between the wheel axle to the sensor.

Figure 8: Distance from axle to sensor
• Velocity $v$ can be measured by making the robot move forward with a certain power while measuring the time and distance. To avoid friction losses set the power to 40 percent of the maximum voltage and let the vehicle move for 3 seconds and measure the distance travelled.

• The last two parameters $T$ and $K$ depend on $G_R(s)$ and how the angular displacement converges over time. With a power of 60 percent of maximum voltage the system response can be illustrated in figure 5. By solving the differential equation $y(T) = 0.63 \times 60K$ so $T$ and $K$ can easily be measured.

![Figure 9: Output when $\Delta u$ is an unit step](image)

**Figure 9:** Output when $\Delta u$ is an unit step

### 3.2.3 Track

![Figure 10: Track with gradient and dashed lines.](image)

(a) ![Track with gradient and dashed lines.](image) (b)

**Figure 10:** Track with gradient and dashed lines.
The track was plotted as an gradient and printed out on an A1-paper see figure 10 a. An IR-sensor was then used on the gradient to set an ideal reference point, usually somewhere in the middle of scale to provide space for the ILC to converge. In part two of the project an estimation of the chassis speed had to be made for the traction control. The speed was calculated in relation to the dashed lines on the track see figure 10 b. Two IR-sensors was used because the vehicle kept missing a few lines in the curves every lap. Due to the speed of the vehicle varies from straights to curves we had to take this into account and use two different algorithms.

**Gradient** The PID controller input is error in millimeter while the light sensor reads the current light value under the vehicle. Therefore measurements of the track is made and a function that describes how the light value and error in millimeters are related.

![Figure 11: The intensity on the gradient from our track.](image)

Using Matlab to find the best fit to the curve seen in figure 11 resulted in a fourth order polynomial, but because of changing light conditions and the sensitivity around the outer values in this graph a simpler and more reliable linearization was made.

**Slip**
Slip is introduced by removing the rubber from the tires. This especially at higher speeds makes the vehicle loose traction very often. In order to add more traction we use friction tape and OH paper placed around the track.

### 3.3 ILC

When choosing the parameters in an ILC-controller the choice is not obvious. There are no clear ways of choosing the parameters or the implementation, many systems require the use iterations and tuning to be able to achieve optimal behaviour form the controller [4]. In this case two implementations were considered, a pure proportional feed forward and a PD feed forward both using the case of causal and non-causal feed forward. This can be seen in Fig. 12. The choice was made to go with the proportional causal controller after
some preliminary tests since the likelihood of the any other controller performing better was deemed very small. The RMS achieved with the proportional ILC also was within the accuracy of the sensor used to measure it so any improvement would be very hard to measure.

![ILC system diagram](image)

Figure 12: ILC system

### 3.4 Traction Control

The traction control problem is solved by scaling the input signal $\Delta u$ when we lose grip. The preset speed, $u_0$, also needs to be regulated in order to be able to turn appropriately. As mentioned in the theory the strategy is to compare the driven wheel speed with the chassis speed in order to determine the slip. In this case it is not possible to determine the rotation of the non-driven wheel so other methods had to be used. The following strategies where proposed:

1. Accelerometer
2. Distance measurements and sample time
3. Kalman filter using the sensors of both the above and driven wheel speed as inputs.

Since an accelerometer was missing at the start of the project the distance measurements was chosen and the map altered so measurements could be made. The inside of the track was marked with 123 equidistant\(^1\) dashed lines which was measured by a set of light sensors. In order to make the speed estimation more robust several samples was used to calculate the speed. The measurement will not be as instantaneous as if only one sample was used but the error in estimation produced by the turning of the vehicle had less impact on the speed estimation. Two different algorithms had to be used depending whether the vehicle was driving on the straight part of the track or turning. To solve this another light sensor was used and four markings along the outside of the track. Calibrating the chassis speed with the rotational speed of the tires with rubber where there was no detectable slip and setting the slip threshold. If the slip threshold was reached the speed would be scaled down according to the formula below. $\xi$ is the antislip coefficient.

\[^1\text{The map was created using InkScape.}\]
which is multiplied with the input to the motors.

\[
\lambda \geq 0.15 \Rightarrow \xi = \left| 1 - \lambda \right|
\]

\[
\lambda < 0.15 \Rightarrow \xi = 1
\]  

(11)

Figure 13: Traction control system

3.4.1 Introducing slip

In order to introduce slip the rubber on the tires where removed. Tests where made with different base speed, \( u_0 \). Changing the radius of the wheels and the speed introduces a model error of our system. Attempting to configure a PID with the new system resulted in a divergent behavior because of the non-linearities but the old one was still able to stay on the track. The error however was significantly larger than before, which is to be expected. The vehicle only needed to stay on track so the same PID tuned with the tires was used with the new track conditions. To introduce different friction along the track friction tape was used placed on one of the straight part of the tracks.
### 3.4.2 Calibration of slip and slip threshold

When calibrating of the estimated slip a comparison was made with a car with and without rubber on the wheels. Letting the signal pass through a low pass filter trends showed slip both with and without the rubbers. The result in figure 14 shows this. The slip $\lambda$ was chosen to be greater or equal to 0.15 in order to let the vehicle maintain the current speed if the slip is smaller or equal to using rubber tires with $u_0$ set to 40%.

![Slip on the same track with or without tires](image)

Figure 14: To introduce an difference in slip, the same tyres was used with and without rubber. The maximum voltage was set to 40%.
4 Result

4.1 ILC implementation

The implementation of the PD-type ILC-controller was interrupted by a search for faults in the PID-controller. Another problem that was discovered was the drift in the ILC due to temporal sampling. This time lost resulted in the use of a first order proportional ILC-controller and a new track with spacial sampling. After controlling convergence over 30 laps for several different proportional gains, shown in Fig. 15, the best results was achieved by a 0.6 proportional gain. The difference in RMS goes from 3.5mm to 1.5mm which is an improvement of 57% from using only the PID.

![Figure 15: The RMS error for different proportional gains in the first order proportional ILC, the values are splined to improve quality](image)

As seen in Fig. 16a the clear periodicity in the signal, which occurs from the curves, vanishes after about 7 laps with the ILC turned on but without it there is not much change.
(a) Error, $y_{ref} - y$, passed though a low-pass filter with cut-off freq. 0.1324Hz, for PID only and ILC

(b) RMS error of only PID and first order proportional ILC with gain 0.6, the values are splined to improve quality

Figure 16: The results with the best improvement using a first order proportional ILC

4.2 Traction control

The traction control experiments was performed using different base speeds. The results shown in figure 17a and 17b the slip before and after traction control is turned on after 10 laps. With the higher speed it is clear improvement but not with the lower speed. This is because of the slip threshold $\lambda$ is set to 0.15.

Figure 17: Calculated slip with an maximum voltage of 50 and 70 percent see figure 17 a and b. Traction control is enabled after 10 laps and the signal is low-passed filtered in Matlab to highlight the trends.
Figure 18: Estimated wheel and chassis speed with a maximum voltage of 50 and 70 percent see figure 18 a and b. Traction control is enabled after 10 laps and the signal is low-passed filtered in Matlab to highlight the trends.

Figure 19: Root mean square error with and without traction control. Figure 19 indicates no improvement in the ability to hold the line with traction control neither with $u_0$ at 50% and 70%

4.3 ILC and Traction Control

Tests also were performed using only ILC in low friction case and both ILC and Traction Control. The results can be seen in figure 20. It should be noted the the robot when off the track when no traction control was used but with traction control it was able to stay on the track.
Figure 20: Root mean square error with and without traction control enabled. With the traction control off, the vehicle starts to have problem to stay on the track.
5 Discussion

5.1 ILC-Algorithm

Figure 15 shows an overview of the result with the ILC-algorithm enabled. The first lap is used to map the error which means that the ILC starts to work first after one lap, that’s why it is zero in RMS error the first lap. In theory it takes a few laps for an ILC-algorithm to kick in and start to work, this means that the error will increase the first laps before converging see figure 2. This had to be taken into account when choosing the value of $K_p$, it was important not to affect the signal to much and have an modest step of the convergence curve. With an low value of $K_p$ see figure 15, the green $K_p = 0.05$ and red curve $K_p = 0.08$ clearly shows that the ILC-algorithm doesn’t reach the same level of RMS error compared with the black curve $K_p = 0.6$. To further increase the value to $K_p = 0.8$ gave an curve with higher slope but unstable behavior after 4-5 laps.

5.2 Traction control

To introduce slip the tires where removed from the vehicle and the base speed $u_0$ was increased. This means an introduction of an error in the system model which was used to decide the PID controller parameters. In an attempt to create a new PID for the new system the PID became to powerful and could not handle the non-linearities introduced by the slip and the vehicle achieved an unstable behaviour. It might have been possible to lower the parameters and thereby getting a slower controller but the original PID stayed on track so in order to save time this was used for these experiments as well.

It is clear that the control system works as expected at the higher base speed $u_0 = 70\%$. This can be seen in figure 17b where the slip is lowered instantaneously as the traction control is turned on after 10 laps. A very positive result was that the chassis speed was not altered at higher speeds. Looking at figure 18b it is clear that both the wheel speed is lowered but the chassis speed stays the same. In this aspect it is obvious the system is performing better with traction control. However does not improve the ability to follow the line as we see in figure 19. This might have been better if the PID controller was configured after the new system with smaller wheels and a faster base speed but since that system was not able to stay on track without traction control this comparison was never made.

The biggest problem was determining the chassis speed and calibrating this. We see a confusing result from this in figure 17a where the chassis speed is larger than the wheel speed the whole time. However the calibration technique used gave good reliable results on all other experiments and therefore was not recalibrated further. As discussed by Gustafsson in [6] sensor fusions between among others base speed (torque), measured distance and an accelerometer be used as states in a kalman filter.

Normally these systems are used in different settings but an attempt to combine the two was made. The result is shown in 20 and shows that the ILC does not work with the introduced slip and the introduced model error. Instead this diverges which can be explained by realizing that the ILC is linear and does not handle the non-linear slip very well. But these non-linearities has less effect if traction control is used and therefore the ILC is slightly
improves the performance when traction control is turned on. If a two input system would be used for the ILC controller where the other input would be the anti-slip coefficient from previous laps the ILC is likely to perform even better. This was never attempted because time did not allow.

5.3 Programming issues

The main issues with the first iteration robot that was built was the fact that the servos were put on the robot backwards resulting in them having to run backwards. This resulted in much confusion because everything that was programmed had consider the fact that the motors run backward with negative angle readouts.

Using the LeJOS library had many advantages such as not having to care about the architecture of the NXT brick. But it also means that there is no transparency in what is actually happening in the code when using the libraries. This drove the efforts in to making custom classes for PID and modelling. This might not have been optimal since it was a source of doubt in the code when searching for bugs. A stricter plan when coding and organizing the coding might have been useful.

When modelling the system a combination of data from LeJOS tools and Matlab was used. This because PID-tuning tools existing in Matlab. This was very tedious and time consuming when changing the system since an export of data was needed every time. There might be of interest to find a smoother solution.

The same problems arose when using low-pass filters because of noisy sensors. To change the cut-off frequency of the filter required data to be transferred between programs which was time consuming. A solution might be to use Matlab for the project or to use Matlab during identification.
6 Conclusion

We have improved the performance of the racer by a decrease of the RMS with 57\% by using a first order proportional ILC implementation compared to when using just our tuned PID regulator. From this it’s clear that the ILC works. To get even better result one could try higher order ILC algorithms with derivative action. In our case we never tried this because we suspected that the sensor noise was too big to improve the results further because we see the periodic behavior from the curves of the track is diminished.

The traction control was able to make the racer more stable and from just observing the racer we could notice less slip and from the noise level it was clear that the torque was properly adjusted. For future project this part should be an on project so that a better velocity estimation is highly recommended, a sensor merge is suggested. Also obtaining a current meter to measure the input into the system to get a proper torque estimation. With that information more sophisticated traction control systems can be built.

7 Work distribution

- **Linus Wågberg**  
  Head of implementation of algorithms in Java.

- **Lars-Gunnar Olofsson**  
  Head of Traction control and ILC algorithm implementation approach.

- **Erik Forslid**  
  Head of Matlab PID tuning and filter design as well as plotting.

- **Mikael Sahlin**  
  Head of system identification and and modelling.
8 References


[5] Dejun Yin and Yoichi Hori *A Novel Traction Control for Electric Vehicle without Chassis Velocity*

[6] Fredrik Gustafsson *Sensor fusion for accurate computation of yaw rate and absolute velocity, 2001*