Application of an Adaptive Iterative Learning Control Algorithm to a Linear Motor

Semester Project

Bastjan Prenaj

Lausanne, June 2007

Swiss Federal Institute of Technology Lausanne
School of Engineering (STI)
Automatic Control Laboratory

Mark Butcher, Ph. D. Supervisor
Dr. Alireza Karimi
Prof. Roland Longchamp

Contents

Abstract

The Adaptive Iterative Learning Control Algorithm in [1] was implemented and then applied to a linear motor system. The difficulties in the implementation and the design of the parameters are being highlighted and a performance evaluation in comparison with a Classical ILC Algortihm in simulations and experiments is given. The results show that the algorithm gives a better tracking in simulation than the Filtered ILC we used for comparison. In the experiments we achieved to fulfill the constraints given by ETEL. The tracking shows that the algorithm produces oscillations in the steady state which might not be negligible in industrial applications.

Chapter 1

Introduction

Many practical industrial systems, e.g pick and place applications, are operated in a repetitive fashion. Iterative Learning Control (ILC) is a well-established method that takes advantage of this repetitiveness by using the information rich error signals of the past iterations. ILC will thus iteratively generate an input signal that will improve the tracking of a reference or the compensation of a repeating disturbance.
As an example of ILC one could take the way in which a baby learns to walk (Fig. 1.1). A normal baby finds usually after a lot of trials a good way to use its muscles. The convergence of the muscle activation to a useful motion profile will be quite time consuming though, as the baby does not know in advance what a good muscle motion profile could be.
Let us suppose now that there is a genius ILC-Baby that already knows at which strength it has to activate its muscles during a walk. But it does not know what the neuronal excitation signal (the input signal) should look like to get the desired muscle movements (the reference output). So the first trial of the ILC-Baby will probably end in a failure and it will fall down. But during the trial it will have continuously registered the error between the desired and the actual muscle activation. As we already assumed that our ILC-Baby is quite clever, it will use these information rich errors to calculate a better neuronal excitation profile that will reduce the errors for the next trial. So it will stand up and give it a try with the changed excitation profile in its mind. In this manner it will iteratively get better results until it will not fall down anymore.
The basic idea of this example is, that the learning is done in the iteration domain. The input is not being changed during the trial (which is therefore open-loop controlled in the time domain).

A standard ILC updating rule has this form:

$$u_{k+1}(t) = L(q)u_k(t) + P(q)e_k(t)$$

The new input $u_{k+1}(t)$ is calculated using the past input $u_k(t)$ and the past error $e_k(t)$.

However, the performance of ILC will be reduced if there are non-repeating disturbances

Figure 1.1: An example for Iterative Learning: A Baby learning to walk [11]

to the system or noise in the measurement. As the algorithm takes the measured error of the past iteration to find the new input signal, it will in this case anticipate something that will not occur.
Applied to the previous example one could imagine that there is a slight wind coming from the left side, so the ILC-Baby will in this case have a drift to the right side. As it learns from its errors, it will try to compensate the force acting from the left side in the next trial. But if we assume that the wind is gone in the next iteration, i.e the wind was a non-repeating disturbance, the ILC-Baby will compensate a force that is not there any more and consequently drift to the left side.
There are different solutions to reduce the influence of disturbances. Some ILC algorithms and the way they deal with this problem are presented and compared in [4].

The aim of this report (as described in the project definition)

"To implement the method proposed in 'An Adaptive Iterative Learning Control Algorithm With Experiments on an Industrial Robot'[1] first in simulation, and then on a linear motor system.
The proposed method uses Kalman filtering techniques to estimate the system's position, despite the presence of measurement noise. Additionally it is adaptive and therefore gives the ability to take, inevitable, modeling errors into account. In order to achieve rapid convergence of the error from one iteration to the next a quadratic criterion is defined, and the control signal to be applied at the next iteration is calculated in order to minimize this criterion."

This report is organised as follows. In chapter 2 we develop the Adaptive ILC Algorithm proposed in [1] and transform the equations given into the desired tracking formulation. In chapter 3 we explain the different steps of the implementation of this algorithm in Matlab and the solutions for the obstacles that were found. Chapter 4 shows how the different design parameters, that have to be set for the Adaptive ILC Algorithm, have been identified. In chapter 5 we compare the Adaptive ILC Algorithm with a Classical ILC Algorithm (Filtered ILC) in simulations and then in experiments on a linear motor system. Finally in chapters 6 and 7 some conclusions are made and some suggestions for future work on this subject are given.

Chapter 2

Development of an Adaptive Iterative Learning Control Algorithm

In this chapter we will first show the advantages and inconveniences of the Adaptive ILC Algorithm proposed in [1]. In the second part we will derive the reference tracking formulation which we used for the simulations and the experiments.

2.1 Why an Adaptive ILC?

As discussed in the introduction measurement noise and non-repeating disturbances pose a problem for the ILC. The Adaptive ILC reduces the sensitivity to their presence by estimating the controlled error $\epsilon_k(t)$ (see Fig. 2.2) using a Kalman filter in the iteration domain. That way we can avoid (or at least reduce) the influence of the measurement noise. Furthermore we introduce a model uncertainty which takes the modelling errors into account. So the algorithm works also when the system is not perfectly known.
But there is a catch to it. The Adaptive algorithm needs some additional parameters to be set. This is a major inconvenience since for each of the parameters there is a considerable margin of possible values. So the design process can be time-consuming in the experimental phase.

2.2 Transformation into the tracking formulation

In order to bring the Adaptive ILC Algorithm into the desired form for our experiment, we had to transform the presented disturbance rejection formulation in [1] into the reference tracking formulation.
In the disturbance rejection formulation the output $z_k(t)$ is minimized (see Fig. 2.1). Whereas the goal in the tracking formulation becomes to minimize the tracking error $\epsilon_k(t)$. Basically the structure of the algorithm stays the same. We simply introduce the desired reference signal y_d as an additional repetitive 'disturbance'. Trying to compensate the disturbance, the ILC algorithm will then automatically improve the tracking of the reference.

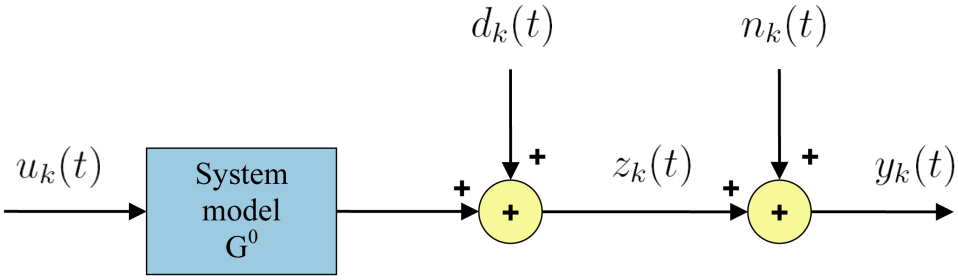

Figure 2.1: Model for the disturbance rejection formulation: A system with two disturbances $d_k(t)$ and $n_k(t)$ acting on the output of G^0.

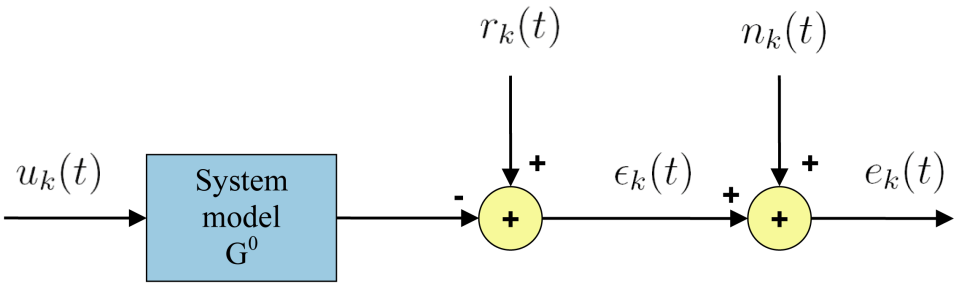

Figure 2.2: Model for the reference tracking formulation: A system with two disturbances $r_k(t)$ and $n_k(t)$ acting on the output of G^0. We included the reference signal y_d in the disturbance $r_k(t) = d_k(t) + y_d$.

The output $y_k(t)$ becomes the measured error $e_k(t)$. The system model G^0 changes its sign $G^0_{tracking} = -G^0_{disturbance\,rejection}$ (Based on how you chose the model for the tracking formulation there are different ways to realize the tranformation. Here we refer our definition to the model shown in Fig. 2.2.)
The following equations are meant to give an overview of the changes in the tracking formulation. A more detailed derivation of the Adaptive ILC Algorithm can be found in [2, chapter 14].
For a better comparability of the two formulations, the following equation numbers correspond to the equations in [1] (which are derived in the disturbance rejection formulation).

The system description for the tracking formulation in matrix form

$$\begin{aligned} \epsilon_k &= -G^0 u_k + r_k \\ e_k &= \epsilon_k + n_k \end{aligned} \tag{2.1}$$

with the disturbance including the reference signal

$$r_{k+1} = d_k + \Delta_{d_k} + y_d. \tag{2.2}$$

Using the updating formula for the disturbance r_k from (2.2) and a model G with a relative model error model Δ_G

$$G^0 = G(I + \Delta_G), \tag{2.3}$$

it is possible to rewrite (2.1)

$$\begin{aligned} \epsilon_{k+1} &= \epsilon_k - G(u_{k+1} - u_k) - \underbrace{G\Delta_G(u_{k+1} - u_k) + \Delta_{d_k}}_{=:\,disturbances} \\ e_k &= \epsilon_k + n_k. \end{aligned} \tag{2.4}$$

The last part of (2.4) is considered as an unknown disturbance. A linear estimator is therefore introduced

$$\hat{\epsilon}_{k+1} = \hat{\epsilon}_k - G(u_{k+1} - u_k) + K_k(e_k - \hat{\epsilon}_k), \tag{2.5}$$

where K_k is the gain of the estimator. By applying standard Kalman filtering techniques (see [2, 14.1.2] and [9]) the estimation procedure becomes

$$\hat{\epsilon}_{k+1} = \hat{\epsilon}_k - G(u_{k+1} - u_k) + K_k(e_k - \hat{\epsilon}_k) \tag{2.6a}$$

$$K_k = P_k(P_k + \hat{R}_{n,k})^{-1} \tag{2.6b}$$

$$P_{k+1} = P_k + \hat{R}_{\Delta_d,k} - P_k(P_k + \hat{R}_{n,k})^{-1}P_k \tag{2.6c}$$

Here it is assumed that both Δ_{d_k} and n_k are uncorrelated from one iteration to the next (i.e. white and gaussian). $\hat{R}_{\Delta_d,k}$ and $\hat{R}_{n,k}$ are the estimates of the true covariance matrices.

To get a trade-off between rapid convergence and excessive input magnitude an LQ-Criterion for optimal control of (2.1) is chosen as

$$J_k = \epsilon_k^T W_\epsilon \epsilon_k + u_k^T W_u u_k. \tag{2.7}$$

It is possible to find an optimal input u_{k+1}^* to the system, by minimizing (2.7)

$$u_{k+1}^* = ((G^0)^T W_\epsilon G^0 + W_u)^{-1} (G^0)^T W_\epsilon r_{k+1}. \tag{2.8}$$

If $W_u = 0$ and r_{k+1} is known (i.e. we have a theoretical feedforward from the disturbance r_{k+1})

$$u_{k+1}^* = (G^0)^{-1} r_{k+1} \tag{2.9}$$

In practice we have to use a system model G instead of G^0. Therefore we get

$$u_{k+1} = (G^T W_\epsilon G + W_u)^{-1} G^T W_\epsilon \hat{r}_{k+1}. \tag{2.10}$$

With $\hat{r}_{k+1}$ being an estimate of r_{k+1}

$$\hat{d}_{k+1} = \hat{\epsilon}_{k+1} + G u_{k+1}, \tag{2.11}$$

we can simplify (2.10) to

$$u_{k+1} = W_u^{-1} G^T W_\epsilon \hat{\epsilon}_{k+1}. \tag{2.12}$$

If we put (2.12) into the observer in (2.5) we get

$$\hat{\epsilon}_{k+1} = \hat{\epsilon}_k + (I + G W_u^{-1} G^T W_\epsilon)^{-1} \kappa_k (e_k - \hat{\epsilon}_k). \tag{2.13}$$

Assumptions

Δ_{d_k} and n_k are both assumed to be white stochastic processes in the time domain. Their estimated covariance matrices can therefore be written in the following form

$$\hat{R}_{\Delta_d,k} = \hat{r}_{\Delta_d,k} \cdot I$$
$$\hat{R}_{n,k} = \hat{r}_{n,k} \cdot I,$$

where $\hat{r}_{\Delta_d,k}$ and $\hat{r}_{n,k}$ are scalars. It is also proposed to assume that $\hat{r}_{\Delta_d,k}$ and $\hat{r}_{n,k}$ do not depend on k. It follows that

$$\hat{R}_{\Delta_d,k} = \hat{r}_{\Delta_d} \cdot I$$
$$\hat{R}_{n,k} = \hat{r}_n \cdot I.$$

Furthermore we assume that

$$P_0 = p_o \cdot I,$$

which implies that

$$K_k = \kappa_k \cdot I.$$

The advantage is that we can rewrite the equations (2.6b) and (2.6c) as scalar equations

$$\kappa_k = \frac{p_k}{p_k + \hat{r}_n} \tag{2.14}$$

$$p_{k+1} = \frac{p_k \hat{r}_n}{p_k + \hat{r}_n} + \hat{r}_{\Delta_d} \tag{2.15}$$

As $\hat{r}_{\Delta_d}$ and $\hat{r}_n$ do not depend on k we can find

$$p_\infty = \frac{\hat{r}_{\Delta_d}}{2}\left(1 + \sqrt{1 + 2\frac{\hat{r}_n}{\hat{r}_{\Delta_d}}}\right) \tag{2.16}$$

Adaptivity

Since P_k and K_k do not depend on the measurements, it is proposed to extend the algorithm to have an adaptivity in the definition of the Kalman gain. The solution is to define a new system disturbance $\hat{R}_{\Delta,k}$ which depends on the measurements from the system (and on $\hat{r}_{\Delta_d}$).
The disturbances in (2.4) are being combined to Δ

$$\epsilon_{k+1} = \epsilon_k - G(u_{k+1} - u_k) - \underbrace{G\Delta_G(u_{k+1} - u_k) + \Delta_{d_k}}_{=:\Delta},$$

and an updating equation for the variation of Δ is proposed as

$$\hat{r}_{\Delta,k} = \frac{1}{n-1}(u_{k+1} - u_k)^T \hat{\Delta}_G^T G^T G \hat{\Delta}_G (u_{k+1} - u_k) + \hat{r}_{\Delta_d}. \tag{2.17}$$

The algorithm is therefore really adaptive, since

$$p_{k+1} = \frac{p_k \hat{r}_n}{p_k + \hat{r}_n} + \hat{r}_{\Delta,k}. \tag{2.18}$$

The Kalman gain κ_k depends on p_k and will therefore depend on the measurements, i.e. it is adaptive.
Here again, we assume that Δ can be considered as a white noise and that we can use scalars instead of matrices.
This supplementary adaptive step gives us the possibility to have a system model G that does not fully capture the true system dynamics as the model uncertainty is being integrated in our algorithm.

Adaptive ILC Algorithm in Matrix Form

1. Let $k = 0$, $\hat{\epsilon}_0(t) = 0$ and $u_0(t) = 0$
2. Apply $u_k(t)$ and measure $e_k(t)$
3. Calculate

$$\begin{aligned}
\kappa_k &= \frac{p_k}{p_k + \hat{r}_n} \\
\hat{\epsilon}_{k+1} &= \hat{\epsilon}_k + (I + GW_u^{-1}G^T W_\epsilon)^{-1}\kappa_k(e_k - \hat{\epsilon}_k) \\
u_{k+1} &= W_u^{-1}G^{-1}W_\epsilon\hat{\epsilon}_{k+1} \\
\hat{r}_{\Delta,k} &= \frac{1}{n-1}(u_{k+1} - u_k)^T\hat{\Delta}_G^T G^T G\hat{\Delta}_G(u_{k+1} - u_k) + \hat{r}_{\Delta_d} \\
p_{k+1} &= \frac{p_k\hat{r}_n}{p_k + \hat{r}_n} + \hat{r}_{\Delta,k}
\end{aligned}$$

4. Let $k = k + 1$. Start next iteration from 2.

Boundedness

In [2, 14.2] an analysis of the ILC is proposed and a sufficient condition for stability of the Adaptive ILC Algorithm (i.e. boundedness of $\hat{\epsilon}_k$) is given

$$\sup_{0<\kappa_k<1} \rho\Big((1-\kappa_k)I - \kappa_k(I + GW_u^{-1}G^T W_\epsilon)^{-1}G\Delta_G W_u^{-1}G^T W_\epsilon\Big) < 1 \tag{2.19}$$

As the model uncertainty $\hat{\Delta}_G$ is only estimated the test is only valid to check if boundedness can be garanteed for this estimated model.
The computation of the matrices gets quickly very heavy. For example in our case the matrix G has a size of 8192x8192 elements.

Chapter 3

Implementation

For the implementation of the Adaptive ILC Algorithm first a transformation from the matrix formulation to one using transfer operators is necessary. The implementation of the latter in MATLAB^{TM} is presented and some solutions for filtering problems that occurred are pointed out.

3.1 Transformation into a transfer operator formulation

The matrix formulation would be too heavy to implement for two reasons:

- The storage of the matrices would take a lot of memory. For our experiments the matrix G would have had 8192x8192 components.
- The calculation with these large matrices is time-intensive.

The solution proposed in [2, 14.1.6] is to use a frequency domain interpretation of the matrix multiplications and to use a transfer operator formulation instead of the matrix formulation. The system description (2.1) then changes to the following form

$$\begin{aligned} \epsilon_k(t) &= -G^0(q)u_k(t) + r_k(t) \\ e_k(t) &= \epsilon_k(t) + n_k(t) \end{aligned} \tag{3.1}$$

which is less general than the matrix formulation since it is only valid for linear time invariant systems.
In this chapter we only present the final solution, the algorithm 14.3 in [2, 14.1.6] (with some small corrections). The setting of the parameters is explained in the next chapter.

Adaptive ILC Algorithm in Transfer Operator Formulation

1. Find $F(q)$ such that

$$F(q)F(q^{-1}) = (1 + \zeta W_u^{-1}(q)G(q)G(q^{-1}))^{-1}$$

2. Let $k = 0$, $\hat{\epsilon}_0(t) = 0$ and $u_0(t) = 0$
3. Apply $u_k(t)$ and measure $e_k(t)$ for $1 \leq t \leq n$
4. Calculate

$$\begin{aligned}
\kappa_k &= \frac{p_k}{p_k + \hat{r}_n} \\
x_{temp}(t) &= F(q)F(q^{-1})\kappa_k(e_k(t) - \hat{\epsilon}_k(t)) \\
\hat{\epsilon}_{k+1}(t) &= \hat{\epsilon}_k(t) + x_{temp}(t) \\
u_{k+1}(t) &= \zeta W_u^{-1}(q)G(q^{-1})\hat{\epsilon}_{k+1}(t) \\
\hat{r}_{\Delta,k} &= \frac{\zeta^2}{n-1}\sum_{\tau=1}^{n}(\Delta_G(q)W_u^{-1}(q)G(q)G(q^{-1})x_{temp}(\tau))^2 + \hat{r}_{\Delta_d} \\
p_{k+1} &= \frac{p_k\hat{r}_n}{p_k + \hat{r}_n} + \hat{r}_{\Delta,k}
\end{aligned}$$

5. Let $k = k + 1$. Start next iteration from 3.

The $F(q)$ is introduced to be able to use the MATLAB$^{\text{TM}}$ command filtfilt.m in the implementation. The definition is the following

$$F(q)F(q^{-1}) = \Big(1 + \zeta G(q)G(q^{-1})H(q)H(q^{-1})\Big)^{-1}$$

Using the fact that $G(q)$ and $H(q)$ can be described as

$$G(q) = \frac{b_G(q)}{a_G(q)}, \quad H(q) = \frac{b_H(q)}{a_H(q)} \tag{3.2}$$

were $a_G(q)$, $b_G(q)$, $a_H(q)$ and $b_H(q)$ are polynomials in q. It is possible to write

$$\begin{aligned}
F(q)F(q^{-1}) &= \Big(1 + \zeta G(q)G(q^{-1})H(q)H(q^{-1})\Big)^{-1} \\
&= \frac{a_G(q)a_H(q)a_G(q^{-1})a_H(q^{-1})}{a_G(q)a_H(q)a_G(q^{-1})a_H(q^{-1}) + \zeta b_G(q)b_H(q)b_G(q^{-1})b_H(q^{-1})}
\end{aligned}$$

The numerator of $F(q)$ is easily found as

$$b_F(q) = a_G(q)a_H(q) \tag{3.3}$$

To find the denominator, $a_F(q)$, first find the roots of

$$a_G(q)a_H(q)a_G(q^{-1})a_H(q^{-1}) + \zeta b_G(q)b_H(q)b_G(q^{-1})b_H(q^{-1}) \tag{3.4}$$

Create a polynomial, $\tilde{a}_F(q)$ having the stable part of the roots of (3.4) as its roots (i.e the roots that correspond to $F(q)$). The unstable roots belong to $F(q^{-1})$. To find $a_F(q)$ the gain of $\tilde{a}_F(q)$ has to be adjusted such that

$$a_F(q) = \gamma \tilde{a}_F(q) \tag{3.5}$$

The γ is found using the following equation

$$\gamma^2(\tilde{a}_F(1))^2 = (a_G(1)a_H(1))^2 + \zeta(b_G(1)b_H(1))^2 \tag{3.6}$$

The sign of the solution is chosen such that $\gamma > 0$.

Notice: The equation (14.34) in [2] should be

$$\hat{r}_{\Delta,k} = \frac{\zeta^2}{n-1}\sum_{\tau=1}^{n}\Big(\Delta_G(q)W_u^{-1}(q)G(q)G(q^{-1})F(q)F(q^{-1})\kappa_k(e_k(\tau) - \hat{\epsilon}_k(\tau))\Big)^2 + \hat{r}_{\Delta_d} \tag{3.7}$$

3.2 Implementation in MATLAB™

The iteration loop of our first solution for the implementation of the Adaptive ILC Algorithm in Transfer Operator Formulation looked like this:

```
kappa(k-1,i)     = p(k-1,i)/(p(k-1,i)+r_n);
xtemp            = filtfilt(bf_q,af_q,kappa(k-1,i)*(e(:,k,i)-eps(:,k,i));
eps(:,k,i)       = eps(:,k,i)+xtemp;
u(:,k,i)         = Zeta*filtfilt(bh_q,ah_q,...
                   flipud(filter(bg_q,ag_q,flipud(eps(:,k,i)))));
r_delta(k-1,i)   = (Zeta^2*sum(delta_g_q*(filtfilt(conv(bh_q,bg_q),...
                   conv(ah_q,ag_q),xtemp)).^2))/N+r_delta_d;
p(k,i)           = (p(k-1,i)*r_n)/(p(k-1,i)+r_n)+r_delta(k-1,i);
```

We used the command "filtfilt.m" to perform the zero-phase filtering and received very high oscillations at the beginning and the end of the filtered signal. The description of "filtfilt.m" in the MATLAB™ documentation explains that the implementation attempts to "minimize these startup transients by adjusting initial conditions to match the DC component of the signal and by prepending several filter lengths of a flipped, reflected copy of the input signal".
In [10, page 169] the authors had a similar problem when implementing infinite time filters on finite data. The solution we chose was to elongate the input signal by reflecting the original input signal on both ends, to filter the extended input signal and to extract afterwards the originally desired filtered input signal. So it is interesting to see, that even though the "filtfilt.m" already includes this technique of reflecting the input signal to minimize the transients, we did not obtain good filtering results until we used our own extension of the input signal.

Chapter 4

Identification of the Design Parameters

In this chapter we give a summary on how to chose the different design parameters for the Adaptive ILC. In addition an explanation on the identification of the system model, the model uncertainty and the noise level can be found.

4.1 Summary of the design scheme given in [1]

LQ design

G Standard system identification

W_u Choose W_u^{-1} as a realization of a zero-phase lowpass filter with cut-off frequency at the desired bandwidth of the ILC algorithm, i.e. $W_u^{-1} = H(q)H(q^{-1})$ with $H(q)$ being a Butterworth filter.

W_ϵ $W_\epsilon = \zeta \cdot I$. The value of ζ will decide how much the ILC scheme should try to resemble the inverse system approach. If ζ is too small the error will not have enough weight in the LQ criterion and the convergence will not be to the desired reference. The experiments showed that if ζ is too big, the oscillations we received had a bigger amplitude. So the choice of ζ is a trade-off between these constraints.

Kalman filter

G see LQ design

$\hat{\Delta}_G$ Standard system identification and model error modeling (see [7])

p_0 This initial choice indicates how well the initial estimate $\hat{\epsilon}_0$ describes the real value. Choose the initial value p_0 large enough, since it will converge to $\hat{r}_{\Delta,k}$ already after one iteration.

$\hat{r}_n$ Choose $\hat{r}_n$ based on physical knowledge, i.e. measure the noise level.

$\hat{r}_{\Delta_d}$ Choose a "right value" for κ_∞. This was an unpleasant part of the parameter design since there are not sharp constraints on the margin.
As an indicator for "good" or "not good" one can take the equation (2.6a). κ_∞ says how much you want to weight your model error in the end. If you trust your model choose κ_∞ small. If don't trust your model, and you want to put some weight on measurement errors, choose κ_∞ bigger.
Given κ_∞ and $\hat{r}_n$ you can calculate $\hat{r}_{\Delta_d}$ by first using the equation (2.15) to find p_∞

$$p_\infty = \frac{r_n \kappa_\infty}{1 - \kappa_\infty}, \tag{4.1}$$

and then the equation (2.16) to find $\hat{r}_{\Delta_d}$

$$\hat{r}_{\Delta_d} = \frac{2 \cdot p_\infty^2}{(\hat{r}_n + 2 \cdot p_\infty)}; \tag{4.2}$$

4.2 Identification

Identification of the system model G

An approximate Box-Jenkins model of the closed-loop motor system was identified using standard identification techniques. We used randomly shifted sinusoids at 6 frequency bands from 1Hz to 1000Hz as input signals. We chose this input signal, because we wanted to excite our system with the maximum possible magnitudes at each band, that still would not saturate the system. By doing so, we would get at each band the best possible quality of the output signal. The magnitudes were identified in experiments for each band.
For the system identification we added all the input signals of the different bands to one input signal containing all 6 frequency bands. This is conform to the superposition principle as our system is assumed to be linear. The output signal for the identification was accordingly the sum of the 6 outputs corresponding to the 6 inputs. The identified model was:

$$\begin{aligned} G(q) = & \frac{0.00013357(q+42.99)(q+2.375)(q+0.7406)(q^2-1.888q+0.9285)}{(q^2-1.271q+0.4327)(q^2-1.887q+0.9284)(q^2-1.508q+0.7072)} \\ & \cdot \frac{(q^2+1.832q+1.06)(q^2-0.3253q+0.6409)}{(q^2+1.783q+0.9537)(q^2-0.2028q+0.6573)} \end{aligned} \tag{4.3}$$

$T_{sample} = 0.0005s$

The order of the model had to be set quite high to satisfy the validation tests (as described in [5],[6],[8]). But since this does not pose a problem for the calculations in ILC, the model was considered to be useful.

Identification of the model error model Δ_G

The identification of the model error model was done according to [7] by using model validation as model error modeling. First we identified the system model G. We simulated the output of our model $y_m(t)$ to the input $u(t)$ and substracted it from the real, experimental output $y(t)$ to get the model error $e(t)$:

$$\begin{aligned} y_m(t) &= G(q)u(t) \\ y(t) &= G^0(q)u(t) + n(t) \\ e(t) &= y(t) - y_m(t) \end{aligned} \tag{4.4}$$

With the definition of the model uncertainty

$$G^0(q) = G(q)(1 + \Delta_G(q))$$

we can simplify the equations 4.4 as follows

$$\begin{aligned} e(t) &= y(t) - y_m(t) \\ &= (G(q)^0 - G(q))u(t) + n(t) \\ &= \Big(G(q)(1 + \Delta_G(q)) - G(q)\Big)u(t) + n(t) \\ &= \underbrace{\Delta_G(q)G(q)}_{\Delta_G(q)'} u(t) + n(t) \end{aligned} \tag{4.5}$$

$\Delta_G(q)'$ is the identified model if we set $u(t)$ as the input and $e(t)$ as the output for the model error model identification. In this case, in order to get the desired $\Delta_G(q)$, we have to use the inverse of our system $G^{-1}(q)$.

$$\Delta_G(q) = \Delta_G(q)'G^{-1}(q)$$

Since in our case $G^{-1}(q)$ has unstable zeros, $\Delta_G(q)$ will be unstable. We found two different approaches to solve this problem:

Solution #1: Stabilize the unstable $\Delta_G(q)$

Proof:

$$\hat{r}_{\Delta,k} = \frac{\zeta^2}{n-1} \sum_{k=1}^{n} (\underbrace{\Delta_G(q)W_u^{-1}(q)G(q)G(q^{-1})x_{temp}[k]}_{=:f[k]})^2 + \hat{r}_{\Delta_d} \tag{4.6}$$

$$\sum_{k=1}^{n} f[k]^2 = \sum_{k=1}^{n} |f[k]|^2 \tag{4.7}$$

with Parseval's theorem:

$$\sum_{k=-\infty}^{\infty} |x[k]|^2 = \frac{1}{2\pi}\int_0^{2\pi} \left|X(e^{j\theta})\right|^2 d\theta \tag{4.8}$$

and the assumption that $f[k]$ is zero for values outside of the interval $[1, n]$ (see remark below for the justification of this assumption) you get:

$$\sum_{k=1}^{n} f[k]^2 \approx \frac{1}{2\pi}\int_0^{2\pi} \left|F(e^{j\theta})\right|^2 d\theta \tag{4.9}$$

$$\begin{aligned}\left|F(e^{j\theta})\right|^2 &= \left|\Delta_G(e^{j\theta})W_u^{-1}(e^{j\theta})G(e^{j\theta})G(e^{-j\theta})X_{temp}(e^{j\theta})\right|^2 \\ &= \left|\Delta_G(e^{j\theta})\right|^2 \left|W_u^{-1}(e^{j\theta})\right|^2 \left|G(e^{j\theta})\right|^2 \left|G(e^{-j\theta})\right|^2 \left|X_{temp}(e^{j\theta})\right|^2\end{aligned} \tag{4.10}$$

with

$$\begin{aligned} a &= |a|\, e^{j\angle a} \\ b &= |b|\, e^{j\angle b} \\ |a \cdot b| &= \left||a|\, e^{j\angle a} \cdot |b|\, e^{j\angle b}\right| = |a| \cdot |b| \end{aligned}$$

Only the magnitude of $\Delta_G(e^{j\theta})$ is of interest for the calculation. Therefore we can stabilize the unstable poles of $\Delta_G(z)$ by reflecting the roots with magnitudes greater than one inside the unit circle. This will only change the phase of the filter. The magnitude stays the same.

Remark: The assumption that $f[k]$ is zero for values of k outside of the interval $[1, n]$ is valid in our case, since we extract only the n values of $f[k]$ that belonged to the original signal to build the sum in (4.6) (after having elongated it for filtering, as described in section 3.2).
For a better readability the $f[k]$ is also named "f" in the MATLAB$^{\text{TM}}$ implementation of the algorithm (see Appendix).

Solution #2: Modify the input for the model error model identification

A simpler solution to the problem consists of modifying the input to the identification, so as to get $\Delta_G(q)$ directly.

$$\begin{aligned} e(t) &= y(t) - y_m(t) \\ &= (G(q)^0 - G(q))u(t) + n(t) \\ &= (G(q)(1 + \Delta_G(q)) - G(q))u(t) + n(t) \\ &= \Delta_G(q)\underbrace{G(q)u(t)}_{=:u_{mod}(t)} + n(t) \\ e(t) &= \Delta_G(q)u_{mod}(t) + n(t) \end{aligned}$$

As shown in the foregoing equations we can avoid now the usage of $G^{-1}(q)$ to get $\Delta_G(q)$. Instead we use the filtered signal $u_{mod} = G(q)u(t)$ as the input signal and the model error $e(t)$ as the output in order to identify the model error model directly.

For the simulations and the results we used the $\Delta_G(q)$ which we identified using the second approach. It had the following form:

$$\Delta_G(q) = \frac{-0.038937(q-0.9959)(q-1.107)(q^2-2.042q+1.065)}{q^8(q^2-0.8295q+0.1914)(q^2-1.847q+0.9376)} \cdot \frac{(q^2-1.447q+1.077)(q^2+1.26q+1.293)}{(q^2-1.516q+0.8171)(q^2+1.661q+0.9476)(q^2-1.093q+0.916)}$$

$T_{sample} = 0.0005s$

Identification of the noise variance $\hat{r}_n$

To identify the noise variance of our system, we put a zero signal as the input to our system and measured the output. The deviations we found were of an amplitude of maximum 8nm. The variances found were in a range of 2 to 330 $[upi^2]$. upi is the increment unit system of the linear motor system. The conversion is defined as follows

$$1\ [m] = 4096 \cdot 10^6\ [upi]$$
$$1\ [upi] = \frac{1}{4096 \cdot 10^6}\ [m]$$
$$\cong 0.244\ [nm]$$

Chapter 5

Results

An analysis of the performance of the Adaptive ILC Algorithm compared to a classical ILC algorithm (Filtered ILC) is given in simulations and experiments. In addition the effort to find the design parameters for the adaptivity is investigated and a comparison with simpler solutions is given.

5.1 Simulations

Fig. 5.1 shows the desired output that was used for performance evaluation in both, simulations and experiments. The same reference was already used in [4] for comparison of different ILC algorithms.
The model G and the model error model Δ_G were taken as identified in the previous chapter. The measurement noise, $n_k(t)$, was taken as a normally distributed, random sequence with $E\{n_k(t)\} = 0$ and $\sigma_d^2 = 100[upi^2] = 6.04e-12[mm^2]$. Based on our design parameters for the Adaptive ILC the likewise normally distributed system disturbance $\Delta_{d_k}(t)$ was defined with $E\{n_k(t)\} = 0$ and $\sigma_d^2 = 2.02[upi^2] = 1.22e-13[mm^2]$.
The further choices for the Adaptive ILC were: For the LQ design: $\zeta = 10^9$, $H(q)$ fifth order Butterworth filter with cut-off frequency at 400Hz.
For the Kalman filter: $\kappa_\infty = 0.1$, $p_0 = 10^{20}$ For each of the following versions of the ILC we carried out 30 iterations. To obtain an estimate of the expected value and the variance of the error at a specific time we repeated each experiment 60 times.
The expectation operator was implemented as follows

$$E\{v_k(t)\} = \frac{1}{N_{exp}} \sum_{j=1}^{N_{exp}} v_k^j(t) \tag{5.1}$$

with $v_k^j(t)$ is an arbitrary signal at time t and experiment j and N_{exp} is the number of experiments.
The variance of the error is defined as follows

$$\sigma_\epsilon^2(t) = E\{\epsilon_k(t)^2\} - E\{\epsilon_k(t)\}^2 \tag{5.2}$$

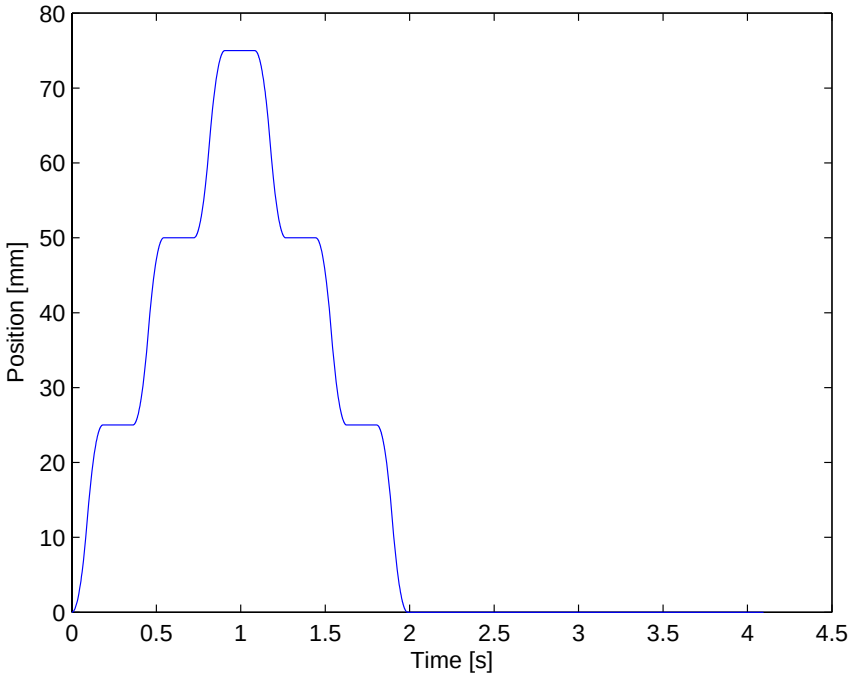

Figure 5.1: Desired output position y_d used in the simulations and experiments

The 2-norm of the error is defined as

$$||\epsilon_\infty||_2 = \Big(\frac{1}{N}\sum_{t=0}^{N-1}\lim_{k\to\infty} E\{\epsilon_k(t)^2\}\Big)^{1/2} \tag{5.3}$$

The results of the simulations that were used to evaluate the advantages of the Adaptive ILC are summarized in Table 5.1.

5.1.1 Comparison with Filtered ILC

For the design of the parameters for the Filtered ILC we used the model (4.3) to calculate a phase lead compensator as an approximation to $\hat{G}^{-1}(q)$. This gave $\hat{G}^{-1}(q) = 0.85q^7$ and the cut-off frequency of 400Hz was chosen for the fifth order Butterworth filter used to assure monotonic convergence.
As can be seen at first sight in Fig. 5.2, the Adaptive ILC gives a much better tracking of the desired output. In a perfect world, where our assumptions are correct (disturbances are white and gaussian) and the model uncertainty is exact, we obtain a fast convergence and a variance of the error $\epsilon_{30}(1.02s)$ that is almost equal to the variance of the load disturbance Δ_{d_k}.

Type	$\lvert E\{\epsilon_{30}(1.02s)(mm)\}\rvert$	$\sigma^2_{\epsilon_{30}}(1.02s)(mm^2)$	$\lvert\lvert\epsilon_{30}\rvert\rvert_2(mm)$
Adaptive ILC	0.46e-06	3.04e-13	8.74e-07
Filtered ILC	1.69e-07	1.01e-12	2.75e-05
Kalman gain κ constant	0.37e-06	4.61e-12	2.07e-06
Simple model error model Δ_G	0.42e-06	3.23e-13	8.80e-07

Table 5.1: Expected value and variance of $\epsilon_{30}(1.02s)$ and $||\epsilon_{30}||_2$ found over 60 simulations

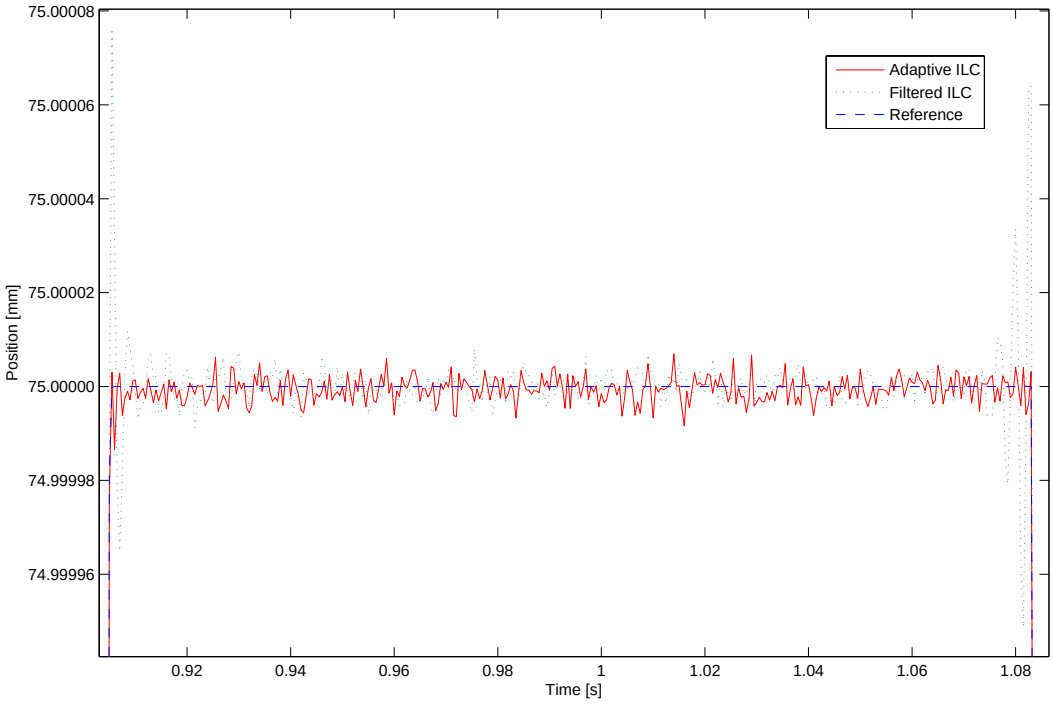

Figure 5.2: Comparison of the tracking output of the Adaptive ILC and the Filtered ILC after 30 iterations $y_{30}(t)$

5.1.2 Comparison with a constant Kalman gain κ

In order to show the effect of the adaptive Kalman gain, we compared the Adaptive ILC with a version of the algorithm where we kept the Kalman gain constant at a level slightly less than one (i.e. $\kappa_{fixed} = 0.99$). In Fig. 5.3 the difference of the root mean squares of the controlled error becomes apparent. The adaptive gain manages to reduce the effect of the measurement noise at higher iterations and gives therefore better results. The values in Tab. 5.1 emphasize this conclusion.

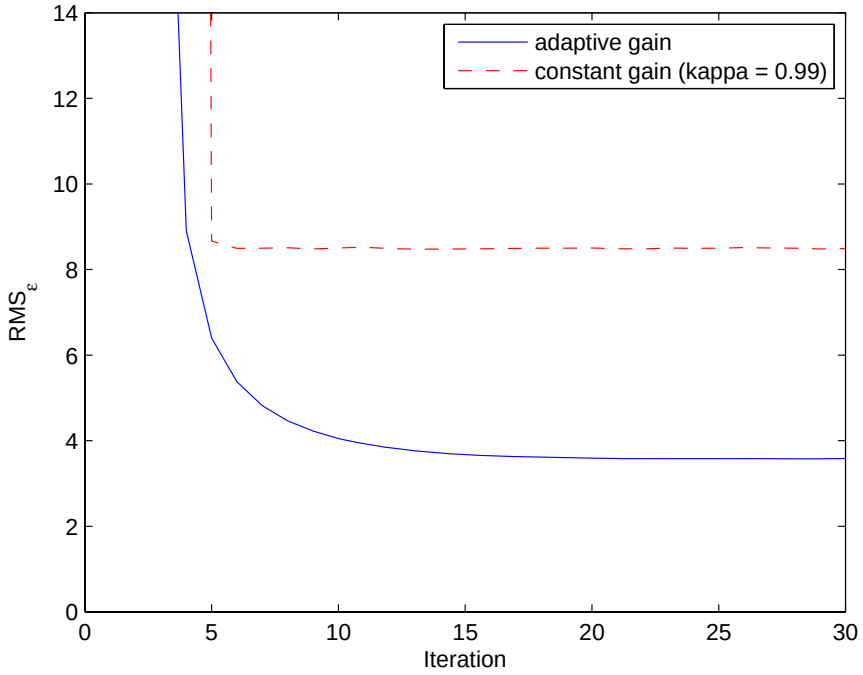

Figure 5.3: Comparison of the root mean square of the controlled error ϵ_k for the Adaptive ILC and the version where κ is fixed at 0.99 (mean over 60 experiments each time)

5.1.3 Comparison with a simpler model error model Δ_G

In [1] a simple model error model of $\Delta_G = 0.5$ is being used for their experiments. In this part of the simulations we wanted to find out if the effort to identify the Δ_G is justified or if a simple gain does the same job.
In the simulations we set the identified model uncertainty as the real uncertainty and defined therefore the real system model following the formula (2.3). So if the model uncertainty is set to $\Delta_G = 0.5$ we are simulating as if the identification of the model error model was imprecise (i.e. too wasteful).
Fig. 5.4 presents the development of the adaptive gain κ for the two versions. The results

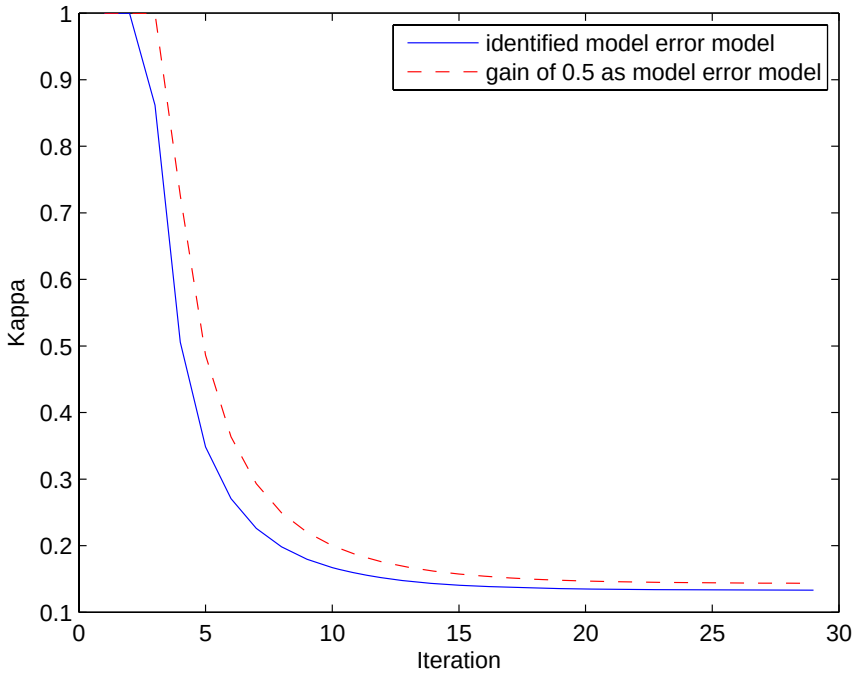

Figure 5.4: Comparison of the adaptive gain κ for the Adaptive ILC with the identified model error model and a simple static gain model error model $\Delta_G(q) = 0.5$ (mean over 60 experiments each time)

in Tab. 5.1 show that the identified Δ_G gives slightly better results. So the model error modeling is justified (at least in the simulations).
It can be seen in the bode diagram of the identified model error model and the simple gain as the model error model (Fig. 5.5), that the magnitude of the complex $\Delta_G(q)$ is smaller for low frequencies. This is because our identified system model is better in this region. At higher frequencies the uncertainty rises and so does the magnitude of the $\Delta_G(q)$.
This comparison demonstrates the usefulness of the adaptivity. Even with a bad model error model we still get reasonable results.

5.2 Experiments

The Fig. 5.6 shows the set up for our experiments. The algorithms were applied to the tracking control of the linear motor system (see Fig. 5.7). The position of the motor used in the experiment is controlled by a two-degree-of-freedom controller operating at a sampling frequency of 2kHz. The motor position is measured by an analog position encoder with a period of 2μm, which is interpolated to obtain a resolution of 0.24nm. The input, $u_k(t)$, computed by the ILC algorithm, is used as the reference signal of the closed-loop system. This means that we do not use the ILC here, as it is proposed in several papers, to

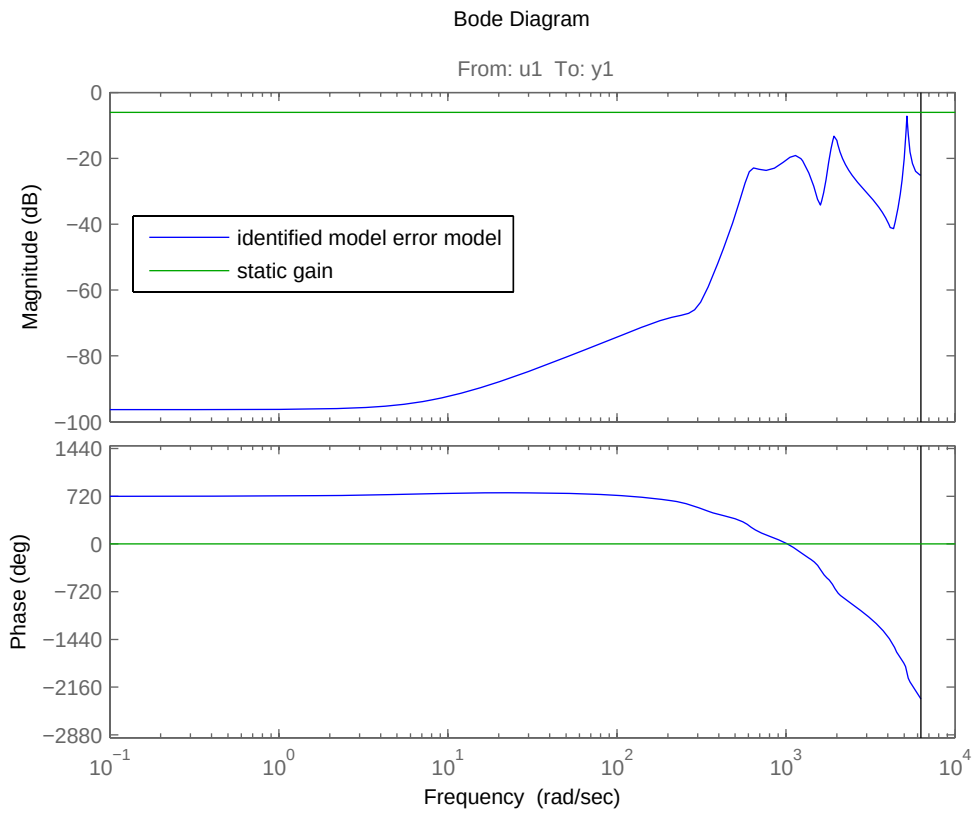

Figure 5.5: Bode diagram of the identified model uncertainty and the simple gain as the model error model

Figure 5.6: Set up used for our experiments: PC (Matlab), Motion Controller and the Linear Motor System (ETEL)

give a feedforward signal. We change the input of our system instead, where the transfer function $G(q)$ represents the closed-loop motor system.

5.2.1 Comparison adaptive vs. decreasing gain

The goal in the experimental phase was to find the best parameters for our Adaptive ILC so as to give the best possible performance. Fig. 5.8 shows a comparison of the output of two experiments with different parameter sets. For the first experiment the set of parameters was exactly the same as in the simulation for the Adaptive ILC. For the second curve the parameters were chosen such that there was no adaptivity anymore in the algorithm, i.e. the κ_k became a simple decreasing gain with $\kappa_k = \frac{1}{k}$. During the tuning of the parameters we discovered that the best results of the Adaptive ILC (i.e. the smallest oscillations) were attained when the Kalman gain was behaving like a decreasing gain. As can be seen in Fig. 5.8 the experiment with the previously identified parameters for the simulation (adaptive gain) gave quite bad results compared to the experiments with decreasing gain.
To have an idea on how the algorithm improves the tracking, Fig. 5.9 shows the developement of the output for the first three iterations using the decreasing gain.

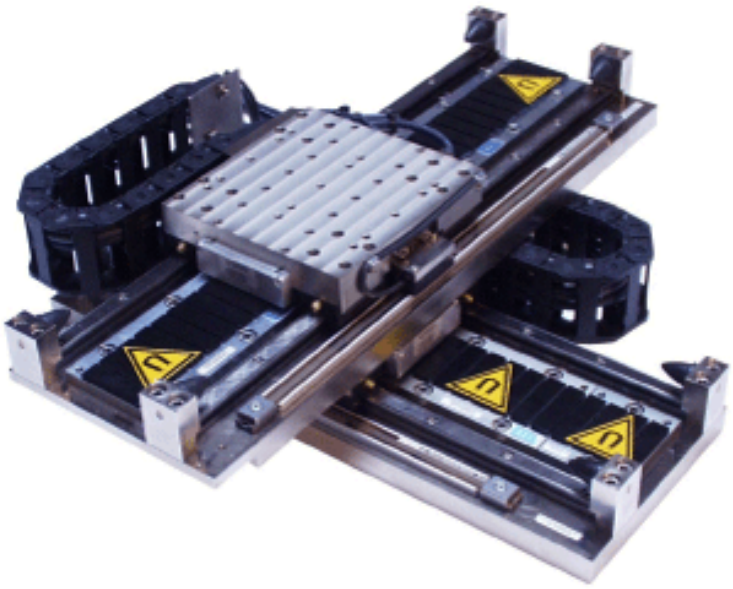

Figure 5.7: Linear motor (ETEL)

Type	$\lvert E\{e_{30}(1.02s)\}\rvert(mm)$	$\sigma^2_{e_{30}}(1.02s)(mm^2)$	$\lvert\lvert e_{30}\rvert\rvert_2(mm)$
Adaptive ILC	4.21e-06	8.63e-11	1.79e-04
Filtered ILC	4.03e-06	7.20e-11	8.62e-05

Table 5.2: Expected value and variance of $e_{30}(1.02s)$ and $||e_{30}||_2$ found over 4 experiments

5.2.2 Comparison with Filtered ILC

As can be seen in Fig. 5.10, the Adaptive ILC shows quite good tracking results in comparison with the Filtered ILC. But there are also oscillations which may be not negligible in the steady state region. However, the magnitudes of the oscillations are not much bigger than the oscillations that were found for the ideal case in the simulations (compare with Fig. 5.2).
Tab. 5.2 shows the values for the expected value and the variance of the measured error at the time $t = 1.02$. Since the values are quite similar, it would be more interesting to know which characteristics of the curves are better (or less damaging) in the real application (e.g. wafer inspection).
The results in Tab. 5.2 can directly be compared with the ones in Tab. 5.1. For the results of the Filtered ILC we received quite similar values in the simulation and in the experiments. For the Adaptive ILC, however, we got a rather worse tracking (e.g. if you take the 2-norm error as a criterion). This can probably be lead back to the fact, that the Adaptive ILC uses several assumptions on the disturbances (that may be wrong) whereas the Filtered ILC has less demands in the parameter design.

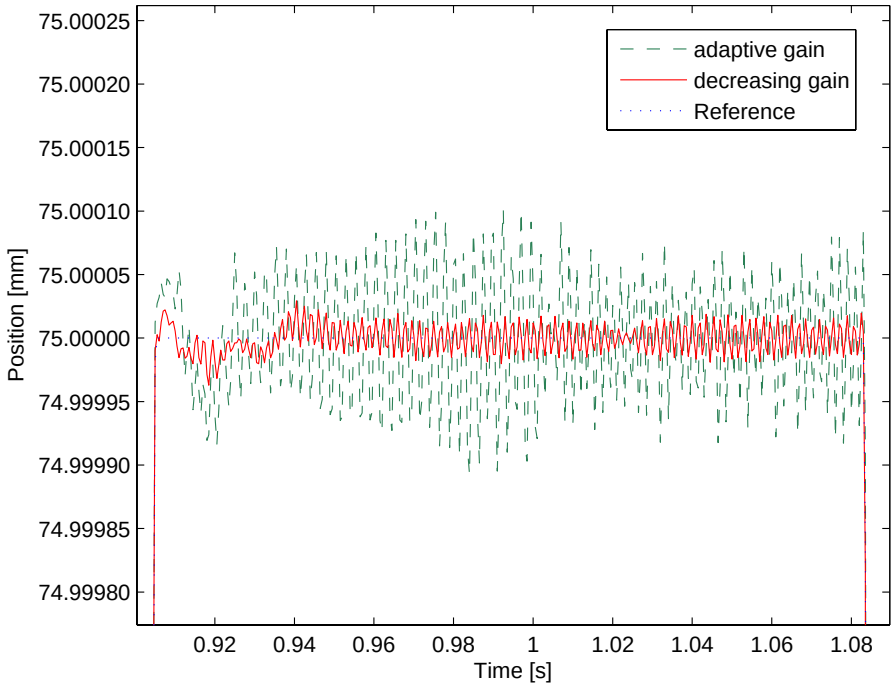

Figure 5.8: Comparison of the tracking if parameters are set as we found them in the simulation (adaptive gain) and the best result we found in the experiments (decreasing gain

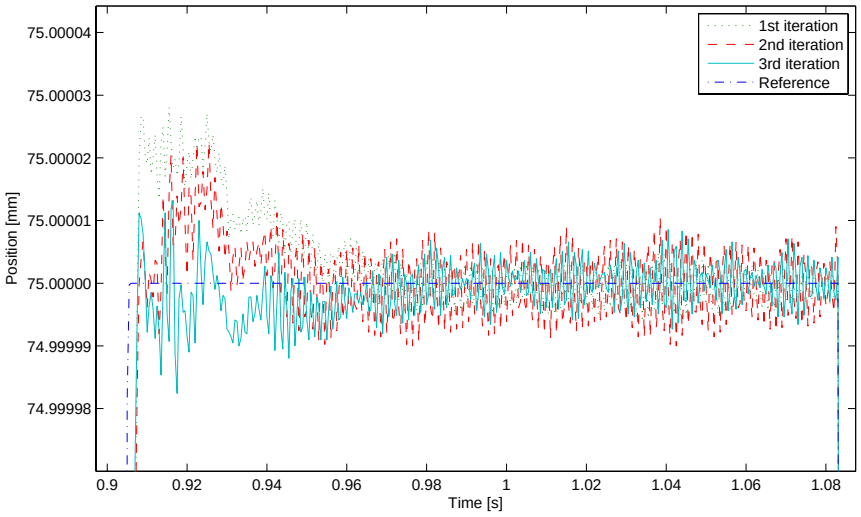

Figure 5.9: Development of the tracking for the first three iterations

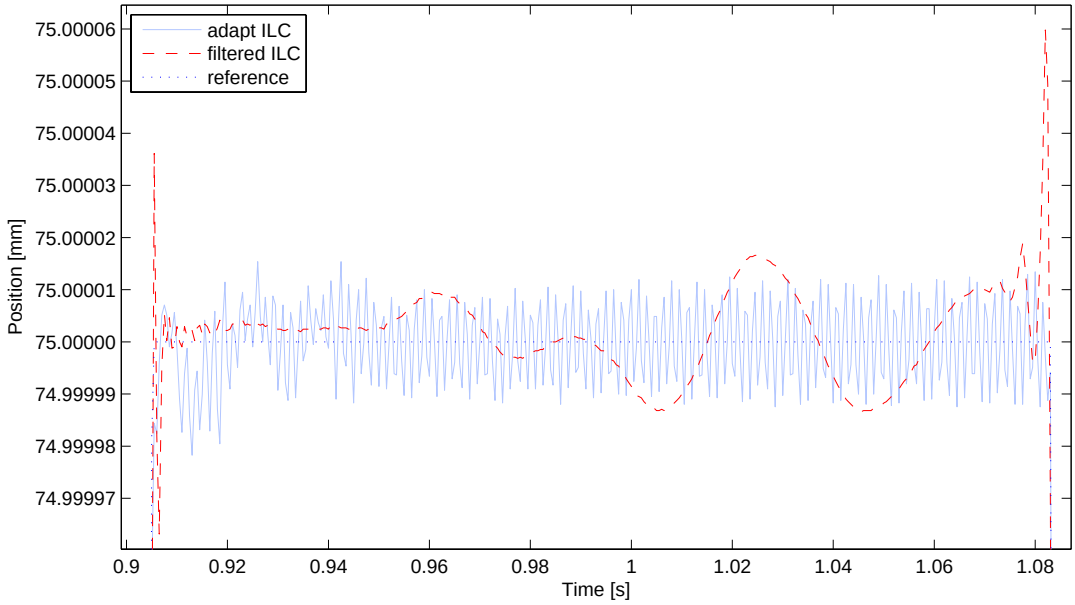

Figure 5.10: Comparison of the Adaptive ILC and the Filtered ILC

5.2.3 Specifications given by ETEL

ETEL, the manufacturer of the linear motor system, gave us the specifications that can be seen in Fig. 5.11. The curve corresponds exactly to the first step of our desired reference for the simulations and the experiments, y_d (see Fig. 5.1). So ETEL specifies a maximum deviation of 100nm from the reference curve from in the interval 0.2s to 0.36s. After 0.36s they want the tracking to have a maximum deviation of 50nm from the reference. The comparison with the output achieved with the Adaptive ILC in Fig. 5.12 shows that the specifications have been fulfilled.

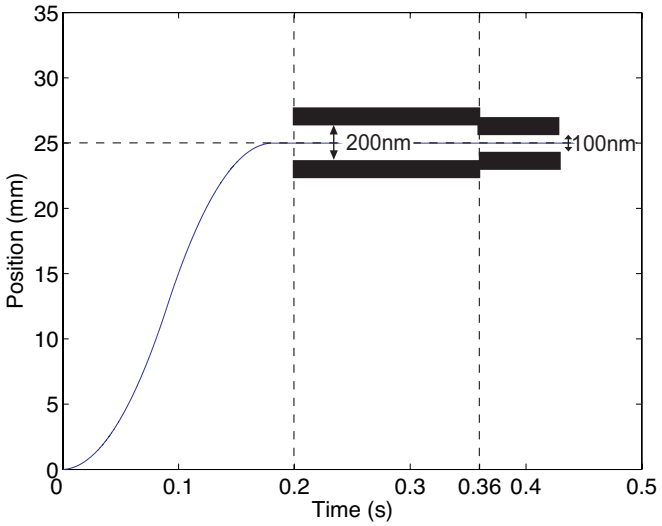

Figure 5.11: Specifications given by ETEL

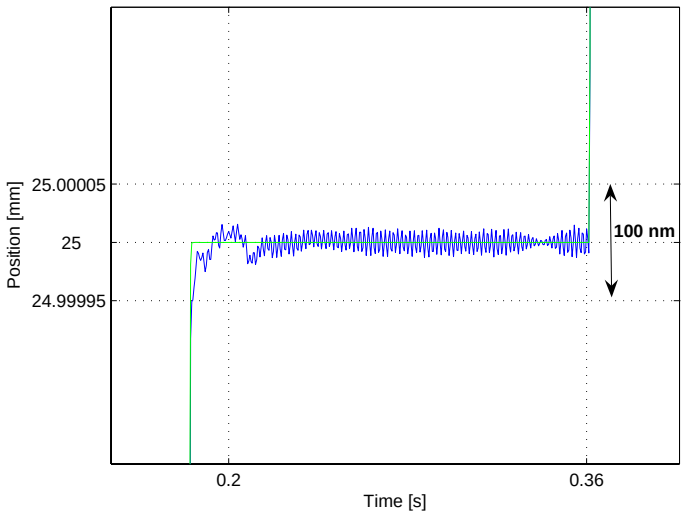

Figure 5.12: Tracking achieved with Adaptive ILC

Chapter 6

Conclusion

The Adaptive ILC gives good tracking results and has a fast convergence. In simulation, i.e. with a perfect knowledge of the system, very good results are attained. So in an ideal world, where we would would know the disturbances perfectly, we have seen that the Adaptive ILC has a better performance than the Filtered ILC.

In the experiments the comparison with the Filtered ILC gives approximately the same size of the errors. More important in this comparison may be the specifications on the output $y_k(t)$ which may have to meet some specifications like highest deviation from reference etc.

However, since the parameter design is time-consuming and the performance in the experiments did not give better results than the comparison with the simpler Filtered ILC, one might prefer the latter for practical application.

The assumptions that the disturbances are white stochastic processes might be too strong and be the reason why we had oscillations in the steady-state regions in our experiments. These oscillations could pose a problem for real applications (e.g. wafer inspection).

Chapter 7

Future work

In order to check if the oscillations are really due to the assumption that the noise is white, one could imagine to develop the Adaptive ILC Algorithm without this simplification and use the matrices ($\hat{R}_{\Delta_d}$, $\hat{R}_n$, K_k, P_k and $\hat{R}_{\Delta,k}$) instead of the scalars.

The amplitudes of the oscillations could pose a problem for real applications (e.g. for wafer inspection). Therefore it would be necessary to find out if these oscillations can be accepted or if the Adaptive Algorithm is not applicable.

Chapter 8

Acknowledgments

I would like to thank my Ph. D. supervisor Mark Butcher for his support and the good ambiance during the project.

My appreciation goes to Dr. Mikael Norrlöf, the author of the Adaptive ILC Algorithm, for his help and encouragement.

Furthermore I am grateful for the ideas and comments of Dr. Alireza Karimi and Prof. Roland Longchamp.

Chapter 9

Appendix

The MATLAB^{TM} files used for the simulations:

adaptILC.m
Implementation of the Adaptive ILC Algorithm

mirror.m
Extension of the signals before the filtering

cleanVector.m
Extracts the original signal out of the extended signal and extends it again with mirror.m (it "cleans" the vector since it chops the transient starts and ends off)

adaptILC_without_CleanVector.m
The usage of cleanVector gave slightly. This m-file contains the old version of adaptILC without the usage of cleanVector.

filteredILC.m
Implementation of the Filtered ILC

adaptILC_showProblemsWithFiltfilt.m
Shows an old version of the adaptILC.m where we hat the transients because of filtfilt.m

adaptILC_with_negativ_G.m
Shows a different implementation of the adaptILC (different setting of the signs)

Bibliography

[1] M. Norrlof, "An Adaptive Iterative Learning Control Algorithm with Experiments on an Industrial Robot", IEEE Transactions on Robotics and Automation, 18(2), pp.245-251, 2002.

[2] M. Norrlöf, Chapters 14-16 of "Iterative Learning Control: Analysis, Design, and Experiments", Ph.D. dissertation, Linköping Studies in Science and Technology, Sweden, October 2000.

[3] D. Bristow, M. Tharayil and A. Alleyne, "A Survey of Iterative Learning Control", IEEE Control Systems Magazine, 26(3), pp.96-114, 2006.

[4] M. Butcher, A. Karimi and R. Longchamp, "A statistical analysis of certain iterative learning control algorithms", Accepted by the International Journal of Control.

[5] L. Ljung, MATLAB System Identification Toolbox Users Guide, Version 4, MA: The Mathworks, 2000.

[6] L. Ljung, "System Identification, Theory for the user", Prentice Hall, New Jersey, 1987.

[7] L. Ljung, "Model Validation and Model Error Modeling", For the Aström Symposium on Control, Lund, Sweden, August 1999.

[8] D. Bonvin, A. Karimi, "Identification de systèmes dynamiques", Lausanne, October 2006.

[9] B. D. O. Anderson and J. B. Moore, "Optimal Filtering". Englewood Cliffs, NJ: Prentice-Hall, 1979.

[10] B. G. Dijkstra, "Iterative Learning Control, with applications to a wafer stage", Delft University of Technology, 2003.

[11] http://kids.sutterhealth.org/images/running%20baby%20on%20hardwood.jpg, June 28/06/2007.