Continuous State Space Q-Learning for control of Nonlinear Systems

ten Hagen, S.H.G.

Citation for published version (APA):

General rights
It is not permitted to download or to forward/distribute the text or part of it without the consent of the author(s) and/or copyright holder(s), other than for strictly personal, individual use, unless the work is under an open content license (like Creative Commons).

Disclaimer/Complaints regulations
If you believe that digital publication of certain material infringes any of your rights or (privacy) interests, please let the Library know, stating your reasons. In case of a legitimate complaint, the Library will make the material inaccessible and/or remove it from the website. Please Ask the Library: http://uba.uva.nl/en/contact, or a letter to: Library of the University of Amsterdam, Secretariat, Singel 425, 1012 WP Amsterdam, The Netherlands. You will be contacted as soon as possible.
Chapter 4

LQRQL for Nonlinear Systems

4.1 Introduction

In this chapter we will study the applicability of linear approximations for nonlinear system. First we will show that other approaches to design controllers for nonlinear systems are often based on local linear approximations. We will rewrite the nonlinear system as a linear system with a nonlinear correction. This allows us to study the effect of nonlinear correction on the estimations of the SI and LQRQL approach as described in chapter 3. We can show that these two approaches estimate the parameters of the wrong function if this correction is not zero.

In a local part of the state space of a smooth nonlinear function, the correction can be assumed to have a constant value. For this situation we will introduce the extended LQRQL approach. In this approach the parameters are estimated of a more general quadratic Q-function, so that more parameters have to be estimated. The resulting feedback function no longer has to go through the origin. Therefore this approach is more suited in a local part of the state space of a nonlinear function.

The effect of the extension of LQRQL is shown in experiments on a nonlinear system. The choice of system was such that we were able to vary the effect of nonlinearity by the choice of the initial state. In this way we were able to show how the extended approach compares with the other two approaches for different sizes of the nonlinear correction. The experiments were performed in simulation and on the real system.

4.2 Nonlinearities

4.2.1 The nonlinear system

In this chapter we will only consider systems that can be represented by first order vectorized difference equations. This is the class of time discrete Markov systems. The systems are described as (1.1)

\[ x_{k+1} = f(x_k, u_k, v_k), \]  

(4.1)
where \( f \) is a functional mapping that maps the present state, control action and noise to the next state value. We will also assume that \( f \) is a smooth continuous differential mapping, for which the gradient to \( x_k \) and \( u_k \) is bounded.

One class of systems that agrees with (4.1) is the class of linear systems according to (3.1). This is an important class. Due to the linearity, the mathematics of analyzing the system's behavior becomes tractable. For a controller design strategy this is important, because this makes it possible to verify whether control design specification are met. Therefore many controller design strategies are based on linear systems.

To illustrate how linear systems simplify the controller design, take a task for which the objective is to keep the state value fixed at a value \( x_s \). The state value does not change if \( x_{k+1} \) equals \( x_k \). This means that there has to exist a constant control action \( u_s \) for which

\[
x_s = Ax_s + Bu_s \tag{4.2}
\]

holds. The state that is unchanged under control action \( u_s \) is given by:

\[
x_s = (I - A)^{-1}Bu_s, \tag{4.3}
\]

where we will call \( x_s \) the set point. This shows that if a solution exists, the state value \( x_s \) is uniquely determined by the control action \( u_s \). It is also important to note that the existence of this solution is completely determined by the parameters of the linear system. So if it exists, it exists for all possible control actions \( u_s \).

The result of the design of the controller is that we get a system with a feedback controller. In chapter 1 we already showed that for a linear system with a linear feedback, the behavior of the closed loop can be expressed using the parameters of the closed loop. If the close loop is stable the system will always end up in the origin. If the closed loop is unstable, the system will never end up in the origin.

If we take a general nonlinear function as in (4.1) then instead of (4.2) we have

\[
x_s = f(x_s, u_s, 0). \tag{4.4}
\]

In this case it is possible to have more solutions. More solutions means that for one \( x_s \), more values of \( u_s \) are possible, but also that for one \( u_s \), more values for \( x_s \) are possible. This implies that the correctness of a control action can only be verified when considering the exact value of the state and control action. The same holds for the equilibrium state when a feedback function \( u = g(x) \) is used. The equilibrium state is the solution to

\[
x_{eq} = f(x_{eq}, g(x_{eq}), 0). \tag{4.5}
\]

Again it is possible that multiple solutions exist. For some solutions the system will approach the equilibrium while for others it will not. The consequence is that the stability of the system also depends on part of the state space. In some parts of the state space the closed loop can be stable, while in other parts it is unstable. It is clear that this will make the design of a controller very complicated. For the linear case it is just a matter of
making the closed loop stable, for the nonlinear case it may also include defining the part of the state space where the closed loop has to be stable.

We can conclude that the main difference between linear and nonlinear systems is that for linear systems the properties are globally valid. They are given by the parameters of the linear system and do not depend on the state value. For nonlinear systems, properties can be only locally valid. They are given not only by the parameters but also depend on the value of the state.

### 4.2.2 Nonlinear approaches

We will give a short overview of some methods for obtaining feedback functions for nonlinear systems. We will also show how these methods rely on techniques derived for linear systems.

#### Fixed Set Point

In (4.4) the noise was set to zero to define the set point \( x_s \) and the corresponding constant control action \( u_s \). In practice however there is always noise. So even if we have a nonlinear system whose state value equals \( x_s \) when \( u_s \) is applied, due to the noise the next state value can be different. Then the system is no longer in its set point, so the state value will change again. For a general function \( f \) this change can be anything and does not have to bring the system back in its set point.

To make sure the system will go back to the set point, a feedback can be added. The input of this feedback is the difference between the state value and the set point. So the task of this feedback is to control its input to zero. To design a linear feedback that will do this, a local linearization of the nonlinear function has to be made around the set point.

If the mapping \( f \) of the nonlinear system (4.1) is continuously differentiable, the system can be assumed to be locally linear around the set point. Let \( \bar{x}_k = x_k - x_s \) and \( \bar{u}_k = u_k - u_s \). Then, (4.1) can be written according to:

\[
\begin{align*}
    x_{k+1} - x_s &= f(x_k, u_k, v_k) - x_s \\
    \bar{x}_{k+1} &= f(x_s + \bar{x}_k, u_s + \bar{u}_k, v_k) - x_s \\
    &\approx f(x_s, u_s, 0) + f_x(x_s, u_s, 0)\bar{x}_k + f_u(x_s, u_s, 0)\bar{u}_k \\
    &\quad + f_v(x_s, u_s, 0)v_k - x_s \\
    \bar{x}_{k+1} &= \hat{A}\bar{x}_k + \hat{B}\bar{u}_k + \hat{B}_v v_k.
\end{align*}
\]

In (4.6) the set point \( x_s \) is subtracted on both sides of (4.1). Then this is expressed in the new state and control action \( \bar{x}_k \) and \( \bar{u}_k \) in (4.7). Note that we assume that \( v \) has zero mean. In (4.8) the mapping \( f \) is replaced by its first order Taylor expansion around the set point, where the mappings \( f_x, f_u \) and \( f_v \) contain the appropriate derivatives. If \( u_s \) is chosen in such a way that \( f(x_s, u_s, 0) = x_s \) and the mappings are replaced by the matrices \( \hat{A}, \hat{B} \) and \( \hat{B}_v \), then this forms the linear system (4.9).
In (4.9) we have a linear system for which a linear feedback can be designed, but care has to be taken. The linear system is only valid near the set point so its properties are not globally valid.

**Gain Scheduling**

The fixed set point approach only gives a feedback that is valid near the set point. To get a feedback function that is valid in a larger part of the state space, multiple linear models can be used. This means that locally linear models are computed for many different set points. As explained in chapter 2 we can form one global nonlinear feedback by combining all the local linear models.

One approach is to form one global feedback where the control action is determined by only one local linear model that is valid at the current state value. This is Gain Scheduling. A possible implementation of Gain Scheduling is to partition the state space and compute for each partition a local linearized model with respect to the center of the partition. These linear models can be used to compute the appropriate local linear feedbacks. The only difference with the fixed set point approach is that now the local linear feedback does not have to result in an equilibrium state around the center of the partitioning. The feedback may change the state of the system to a part of the state space where a different local model will determine the control action.

**Feedback Linearization**

In some cases it is possible to change a nonlinear system into a globally linear system. This is called feedback linearization. The objective is to change the nonlinear system into a linear system of our choice. This is only possible for a restricted class of nonlinear systems for which the mapping $f$ of the nonlinear system in (4.1) can be written as

$$f(x, u, v) = g(x) + h(x)u + v.$$  \hspace{1cm} (4.10)

with $g$ and $h$ mappings of the appropriate dimensions.

Suppose we want to have a linear system with parameters $\hat{A}$ and $\hat{B}$ and control input $\hat{u}$. Then the control action can be computed according to

$$u = h(x)^{-1}(\hat{A}x - \hat{B}\hat{u} - g(x)).$$  \hspace{1cm} (4.11)

where $\hat{u}$ represents a new control action. Applying (4.11) to a nonlinear system with (4.10) will transform the nonlinear system into a linear system:

$$x_{k+1} = g(x_k) + h(x_k)u_k + v_k$$
$$= g(x_k) + h(x_k)(\hat{A}x_k - \hat{B}\hat{u}_k - g(x_k)) + v_k$$
$$= \hat{A}x_k - \hat{B}\hat{u}_k + v_k.$$  \hspace{1cm} (4.12)

The result is a linear system, so the appropriate new control actions $\hat{u}$ can be determined using conventional linear techniques. The choice of the parameters of the new linear system are made to simplify the design of the controller for the new linear system.
It is clear that this approach is only possible for a restricted class of nonlinear systems. Not only does (4.10) have to hold, but also (4.11) has to exist. In case of a scalar state and action it is clear that $h(x) \neq 0$ is a sufficient condition for the existence of (4.11). For more general situations the existence has to be verified using Lie-brackets. For more detailed information about this approach see [37].

The existence of (4.11) is important, but it also has to be found. When the physics of the system is known this can be computed. When the physics is not known (4.11) has to be approximated by a general function approximator. In that case it is no longer possible to guarantee global properties of the system, since it relies on the quality of the approximation. For a continuous time configuration, conditions for sufficient quality of the approximation for feedback linearization can be given [78]. One of these conditions is that the system is excited enough, so that the state space is explored sufficiently.

### Linear Matrix Inequalities

Linear Matrix Inequalities (LMI) techniques [15] are used to analyze nonlinear systems and proof properties, like stability. These techniques are based on the idea that a nonlinear system can be described by a set of linear systems. If we look at only one state transition of a nonlinear system then a linear system can be defined that generates the same state transition:

$$x_{k+1} = f(x_k, u_k) = A(x_k, u_k)x_k + B(x_k, u_k)u_k.$$  \hspace{1cm} (4.13)

Here $A(x_k, u_k)$ and $B(x_k, u_k)$ represent matrices, whose parameters depend on $x_k$ and $u_k$. So for the current state and control action there is a linear system for which the state transition will also lead to $x_{k+1}$.

To say something about the stability of the nonlinear system, all possible values of $x$ and $u$ should be considered. The parameter matrices $A(x, u)$ and $B(x, u)$ for all possible $x$ and $u$ form a set. If all linear systems corresponding to the matrices in the set are stable, then also the nonlinear system must be stable. This is because for any state transition of the nonlinear system there exists a stable linear system that will have the same state transition.

The set can consist of an infinite number of linear systems, making it impossible to prove the stability of all linear systems. One solution is to use a polytope LMI. In the parameter space of the linear system a polytope is selected that encloses the complete set. If the linear systems corresponding to the edges of the polytope are stable then all linear systems within the polytope are stable as well.

Another approach is to use a norm bounded LMI. One stable linear system with parameters $A$ and $B$ is selected and all linear systems are considered to have parameters $\hat{A}$ and $\hat{B}$ plus an extra feedback. The state transitions of the linear systems are described as $x_{k+1} = \hat{A}x_k + \hat{B}u_k + w(x_k, u_k)$, where the vector $w(x_k, u_k)$ represents the extra feedback.

---

1For simplicity we assume there is no noise.
If it can be proven that the norm of $w$ is less than one for all values of $x$ and $u$, then the nonlinear system is stable.

Since LMIs provide the opportunity to guarantee stability for nonlinear systems, they can be used for stability based controller design. The LMI technique is used to specify the set of feasible feedbacks and a controller design approach is used to select one feedback from that set. The drawback of the LMI approaches is that the set of feasible feedbacks depends on the choice of polytope or $A$ and $B$. In case of an unfortunate choice it is possible that many good feedbacks are rejected. In the worst case the set of feasible feedbacks is empty.

### 4.2.3 Summary

We described some approaches to obtain feedback functions for nonlinear systems. Except for the feedback linearization, all were based on local linear approximations of the nonlinear feedback. These approaches assume that the mapping $f$ is available, but when it is unknown the local linear system can be approximated by the SI approach from chapter 3. The LQRQL approach in chapter 3 was also derived for linear systems. We are interested in using LQRQL to obtain a local linear approximation of the optimal feedback function.

### 4.3 The Extended LQRQL Approach

In this section we will first show that the SI and LQRQL approach from chapter 3, do have limitations when they are applied to nonlinear systems. This is especially the case when the estimations are based on data generated in a small part of the state space. We will show that LQRQL can be extended to overcome these limitations, resulting in a solution that is more appropriate in a small part of the state space.

#### 4.3.1 SI and LQRQL for nonlinear systems

The SI and LQRQL approach were derived for linear systems. In order to see how they would perform when applied to nonlinear systems, we write (4.1) as

$$x_{k+1} = f(x_k, u_k, v_k)$$

$$= \tilde{A}x_k + \tilde{B}u_k + (f(x_k, u_k, v_k) - \tilde{A}x_k - \tilde{B}u_k)$$

$$= \tilde{A}x_k + \tilde{B}u_k + w(x_k, u_k, v_k).$$

Equation (4.15) strongly resembles a norm bounded LMI. In our case the mapping $f$ is unknown, so we cannot use (4.15) as a LMI.

The SI and LQRQL approach obtain a feedback based on a generated train set. If we assume that the train set is generated in a small local part of the state space, we can

---

2The linear system (3.1) is a special case for which $w(x, u, v) = v \forall x, u, v$. 

simplify (4.15) by replacing the nonlinear function \( w(x, u, v) \) with its average value for the train set. So we look at the system:

\[
x_{k+1} = Ax_k + Bu_k + w
\]  

(4.16)

where the value of vector \( w \) has a fixed constant value.\(^3\)

We can apply the SI approach, which will result in the estimated linear system: \( x_{k+1} = \hat{A}x_k + \hat{B}u_k \). In this linear system the correction is not present. This implies that \( \hat{A} \) and \( \hat{B} \) are estimated such that the average value of \( w \) is zero. So when the average value of \( w \) is far from zero, the estimated linear system with \( \hat{A} \) and \( \hat{B} \) is not a good approximation of \( f \) in (4.14). The resulting feedback \( \hat{L}_{SI} \) will not approximate the optimal feedback at the part of the state-action space where the train set was generated.

In order to see the consequences of \( w \) on the result of LQRQL, we can look at the resulting linear feedback function \( u = Lx \). When we apply the feedback function \( u_k = Lx_k \) to control (4.16), the equilibrium state \( x_{eq} \) is no longer at the origin. The equilibrium state is given by:

\[
x_{eq} = (I - A - BL)^{-1}w.
\]  

(4.17)

This shows that the equilibrium state depends on \( L \)! The LQRQL approach was derived for a linear system with \( w = 0 \), so the equilibrium state was always in the origin. Only the trajectory towards the equilibrium state can be optimized.

At the equilibrium state in (4.17) the direct costs (3.2) are not zero. So LQRQL applied to (4.16) will not only optimize the trajectory toward the equilibrium state, but also the value of the equilibrium state. If both the equilibrium state and the trajectory towards the equilibrium state have to be optimized, then the feedback function \( u = Lx \) has insufficient degrees of freedom.

### 4.3.2 The extension to LQRQL

Another way to see why the linear feedback does not have enough degrees of freedom is by looking at its approximation of a nonlinear optimal feedback. Let us assume an optimal linear feedback function \( g^* \) as shown in figure 4.1. The local linear approximation according to \( u = Lx \) is not very good. The figure also shows that the addition of an extra constant \( l \) to the feedback function will allow for a better local linear approximation.

If the feedback function \( u = Lx + l \) is used for (4.16) then the equilibrium state is given by

\[
x_{eq} = (I - A + BL)^{-1}(Bl + w).
\]  

(4.18)

Here we see that if the feedback \( L \) is chosen such that the trajectory towards the equilibrium state is optimal, the value of \( l \) can be used to select the optimal equilibrium state.\(^4\) The \( l \)

---

\(^3\)Strictly speaking this does not have to represent a nonlinear system because the value of \( w \) is also globally valid. We can also view (4.16) as a linear system, where the mean of system noise is not zero. The \( w \) then represents the mean of the system noise.

\(^4\)Whether this is possible depends on \( B \) and \( w \).
Figure 4.1. The local linear approximation of the optimal nonlinear feedback function. The solid lines indicate the region of approximation. Because the feedback $u = Lx$ has to go through the origin, the local linear approximation cannot be a good approximation of the nonlinear function. The approximation of $u = Lx + l$ does not have to go through the origin and is able to match the nonlinear function more closely.

can be interpreted as the constant $u_s$ in the Fixed Set Point approach. The difference is that the value of $l$ is not chosen, but optimized using the extended LQRQL approach.

The resulting feedback function of LQRQL is a consequence of the choice of the quadratic Q-function. For the new feedback function, a new Q-function is needed. The Q-function used in the previous chapter was of the form $Q(\phi) = \phi^TH\phi$, where $\phi_k = [x_k^T \ u_k^T]$. This is not a very general quadratic function. A general quadratic function is given by:

$$Q(\phi) = \phi^TH\phi + G^T\phi + c.$$  \hspace{1cm} (4.19)

By including a term with vector $G$ and a constant $c$, any quadratic function can be represented by (4.19). If (4.19) has the optimal parameters $H^*, G^*$ and $c^*$, the greedy feedback can be found by taking the derivative to $u^*$ and set this to zero. This results in

$$u^* = -(H_{uu}^*)^{-1}(H_{ux}^*x^* + G^{*T})$$

$$= L^*x + l^*.$$ \hspace{1cm} (4.20)

This indicates that the Q-function (4.19) will result in the feedback we want. Compared to (3.16) we see that again $L^* = -(H_{uu}^*)^{-1}H_{ux}^*$. This shows that $L^*$ optimizes the trajectory to the equilibrium state. The difference with (3.16) is that a constant $l^* = -(H_{uu}^*)^{-1}G^{*T}$ is added to the feedback function. The purpose of this constant is to determine the optimal equilibrium state. Note that the scalar $c$ in (4.19) does not appear in (4.21), so it does not have to be included in the estimation.

4.3.3 Estimating the new quadratic Q-function

The estimation method for the new quadratic Q-function will be slightly different because more parameters have to be estimated. In the previous chapter the estimation was based
on the temporal difference (5.18). A similar approach will be used for the Q-function according to (4.19). The difference with (3.19) is that \( \phi_k^T = [x_{k+1}^T, Lx_{k+1} + l] \) and the consequences of the noise are represented by \( \nu_k \). Then in the same way as in (3.20):

\[
\begin{align*}
    r_k &= \sum_{i=k}^{\infty} r_i = Q(x_k, u_k) - Q(x_{k+1}, Lx_{k+1} + l) \\
    &= \phi_k^T H \phi_k + G^T \phi_k - \phi_{k+1}^T H \phi_{k+1} - G^T \phi_{k+1} + \nu_k \\
    &= \text{vec}^q(\phi_k \phi_k^T) \text{vec}^q(H) - \text{vec}^q(\phi_{k+1} \phi_{k+1}^T) \text{vec}^q(H) + (\phi_k^T - \phi_{k+1}^T)G + \nu_k \\
    &= \begin{bmatrix}
        \text{vec}^q(\phi_k \phi_k^T - \phi_{k+1} \phi_{k+1}^T) \\
        \phi_k^T - \phi_{k+1}^T
    \end{bmatrix} \begin{bmatrix}
        \text{vec}^q(H) \\
        G
    \end{bmatrix} + \nu_k.
\end{align*}
\]

This again can be used to express the estimation as a linear least squares estimation:

\[
Y_{\text{EX}} = \begin{bmatrix}
    r_0 \\
    r_1 \\
    \vdots \\
    r_{N-1}
\end{bmatrix} = \begin{bmatrix}
    \text{vec}^q(\phi_0 \phi_0^T - \phi_1 \phi_1^T) & \phi_0^T - \phi_1^T \\
    \text{vec}^q(\phi_1 \phi_1^T - \phi_2 \phi_2^T) & \phi_1^T - \phi_2^T \\
    \vdots & \vdots \\
    \text{vec}^q(\phi_{N-1} \phi_{N-1}^T - \phi_N \phi_N^T) & \phi_{N-1}^T - \phi_N^T
\end{bmatrix} \theta_{\text{EX}} + \begin{bmatrix}
    \nu_0 \\
    \nu_1 \\
    \vdots \\
    \nu_{N-1}
\end{bmatrix} \quad (4.26)
\]

\[
X_{\text{EX}} \theta_{\text{EX}} + V_{\text{EX}}. \quad (4.27)
\]

The estimation \( \hat{\theta}_{\text{EX}} = (X_{\text{EX}}^T X_{\text{EX}})^{-1} X_{\text{EX}}^T Y_{\text{EX}} \) gives an estimation of \( \theta_{\text{EX}} \) and because \( \text{vec}^q(H) \) and \( G \) are included in \( \theta_{\text{EX}} \), it also gives the estimation for \( \hat{H} \) and \( \hat{G} \). The \( \hat{H} \) and \( \hat{G} \) can be used in the same way as in (4.21) to obtain the resulting \( \hat{L}_{\text{EX}} \) and \( \hat{l} \).

The absence of the constant \( c \) in (4.25) indicates that \( c \) does not influence the outcome of the estimation. So the actual function that is estimated is

\[
\hat{Q}(\phi) = \phi^T \hat{H} \phi + \hat{G}^T \phi. \quad (4.28)
\]

This is the function we will use as Q-function and we will call this the Extended LQRQL approach. For this function \( \hat{Q}(0) = 0 \), so it is not a general quadratic function anymore. Still this is general enough because the value of \( c \) also does not influence (4.21). The reason that this constant \( c \) can be ignored completely is that it represents the costs that will be received anyway, regardless of the optimization. This indicates that the optimization is only based on avoidable costs. Costs that will be received anyway do not influence the resulting feedback function.

### 4.4 Exploration Characteristic for Extended LQRQL

The estimation (4.27) shows how the parameters are estimated, but does not indicate how well these parameters are estimated. The results from the previous chapter suggest that the correctness of the estimation depends on the amount of exploration used for generating.

---

5 The Q-function depends on \( L \) and \( l \), but for clarity these indices are omitted.

6 Note that if a discount factor \( \gamma < 1 \) is used in (4.22), the constant \( c \) does influence the outcome.
the train set. The exploration characteristic was introduced in section 3.3.4 to present an overview of the quality of the outcome as a function of the exploration. This will now be used to investigate the results of the extended LQRQL approach. Two questions have to be answered:

- Are the extra parameters \( \hat{G} \) estimated correctly?
- How does the estimation of \( \hat{G} \) influence the estimation of \( \hat{H} \)?

This last question is relevant since \( G \) and \( H \) are estimated simultaneously in (4.27).

The estimation for the extended approach includes \( n_x \) extra parameters and can be written similar to (3.58). The linear equation that has to hold is given by

\[
Y_{EX} = \begin{bmatrix} \psi^{xx} & \psi^{ux} \end{bmatrix} \begin{bmatrix} \theta_{xx} \\ \theta_{ux} \end{bmatrix} + V_{EX}. \tag{4.29}
\]

where \( V_{EX} \) represents the noise and \( \theta_g \) represents the extra parameters that have to be estimated. Similar to (3.63) the estimation error of a row of \( \hat{\theta}_g \) can be written as

\[
\hat{\theta}_{g,i} = \frac{\psi_{g,i}^T P_{n_x+n_u+n_t+1} \psi_{g,i}^*}{\| P_{n_x+n_u+n_t+1} \|_2^2} (V_{EX} - \sum_{j=n_x+n_u+1}^{n_x+n_u+n_t+1} \psi_{g,j}^* \hat{\theta}_{g,j} ). \tag{4.30}
\]

The estimations \( \hat{\theta}_{uu} \) in (3.63) and \( \hat{\theta}_{ux} \) change according to

\[
\hat{\theta}_{uu,i} = \frac{T_{uu,i}^T P_{n_x+n_u+n_t+1} (V_{EX} - \sum_{j=n_x+n_u+1}^{n_x+n_u+n_t+1} \psi_{u,j}^* \hat{\theta}_{uu,j} )}{\| P_{n_x+n_u+n_t+1} \|_2^2}, \tag{4.31}
\]

\[
\hat{\theta}_{ux,i} = \frac{\gamma_{ux,i}^T P_{n_x+n_u+n_t+1} (V_{EX} - \sum_{j=n_x+n_u+1}^{n_x+n_u+n_t+1} \psi_{ux,j}^* \hat{\theta}_{ux,j} )}{\| P_{n_x+n_u+n_t+1} \|_2^2}. \tag{4.32}
\]

The main difference here is that there is the extra \( \psi^{ux} \hat{\theta}_g \) in the estimation errors \( \hat{\theta}_{uu,i} \) and \( \hat{\theta}_{ux,i} \). The dependency of the estimation error \( \hat{\theta}_g \) on the exploration is comparable to that of the SI approach, so the influence of the \( \hat{\theta}_{uu,i} \) and \( \hat{\theta}_{ux,i} \) can be neglected. The \( V_{EX} \) is different from \( V_{QL} \) from the previous chapter in the sense that it now includes the value of \( w \). This means that in the expected estimation errors (3.66) the value of \( \sigma_e^2 \) should be replaced by \( \sigma_e^2 + \| w \|^2 \). The consequence is that even for low noise in the system, if there is a large extra \( w \) still a large amount of exploration is required. In other words, the minimum of the Q-function should be included in the area that is being explored.

**Simulation of the Extended LQRQL characteristics**

We did some simulation experiments to show the reliability of the extended LQRQL approach. To be able to show the correctness of the parameter estimation we took as system the model given by (4.16):

\[
x_{k+1} = Ax_k + Bu_k + v_k + w \quad u_k = Lx_k + e_k + l. \tag{4.33}
\]
In (4.33) the \( w \) is constant so it is the same for all time steps. In the experiments we compared the results with the standard LQRQL approach.

We first did an experiment to see whether the correct value of \( l \) can be found. Because all parameters are estimated simultaneously, we also did an experiment to see whether the estimation of \( G \) has any positive effects on the estimation of \( H \). Finally we did an experiment to see whether an arbitrary set point can be learned, when this set point is determined by the direct costs \( r \).

We used the same parameter settings as in the previous chapter:

\[
A = \begin{bmatrix} -0.6 & 0.4 \\ 1 & 0 \end{bmatrix}, \quad B = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \quad x_0 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \quad L = \begin{bmatrix} 0.2 & -0.2 \end{bmatrix}, \quad \sigma_v = 10^{-4}.
\]

The value of \( w \) depends on the experiment. Each experiment was performed 5 times and we took \( N = 30 \) number of time steps.

**Experiment I:** Can a correct value of \( l \) be found? This requires knowledge about the correct value \( l \). In this experiment we took \( w \) equal to zero so that no additional \( l \) is required. We took as initial value \( l = 1 \). The initial value \( l \neq 0 \), so that the estimated \( \hat{l} \) should be zero.

Figure 4.2(a) shows how the value of \( l \) changes. For low values of \( \sigma_e \) the value of \( l \) does not change. If the value of \( \sigma_e \) is larger than \( \sigma_v \) the correct value of \( l \) is obtained.

**Experiment II:** Does the extension of LQRQL improve the results? In this experiment we used \( w^T = [1 \quad 1] \), so that a \( l \neq 0 \) was required. Both the LQRQL and the Extended LQRQL were applied on the same train sets. We asked the following questions: Does the nonzero \( w \) influence the estimation of \( \hat{L} \)? Also does the nonzero \( w \) influence the amount of exploration that is required?

Figure 4.2(b) shows \( \| \hat{L} - \hat{L}^* \| \) for both approaches, where \( \hat{L}^* \) is the optimal feedback computed with (3.5) and (3.4). For low values of \( \sigma_e \) the value of \( \hat{L} \) equals \( L \) in (4.34). The value of \( \| \hat{L} - \hat{L}^* \| \) decreased around \( \sigma_e \approx 1 \), which is about the same scale as \( w \). We did a similar experiment using \( w^T = [10 \quad 0] \), and there this happened at \( \sigma_e \approx 10 \). This indicates that the sufficient amount of exploration depends on \( \| w \| \) if \( \| w \| > \sigma_v \). Figure 4.2(b) shows that the improvement for the extended approach requires less exploration. For high values of \( \sigma_e \) the value of \( l \) becomes \(-0.5\) and this makes the total costs for the extended LQRQL approach lower than the total costs of the standard LQRQL approach.

**Experiment III:** Can the set point be learned? The set point was introduced in section 4.2.2 to indicate a desired equilibrium state. So learning the set point means optimizing the stationary behavior of the system. The extension of LQRQL was motivated by the inability of LQRQL to deal with the simultaneous optimization of the transient and stationary behavior of the system. We did this experiment to see whether this is possible using the extended LQRQL. We changed the costs in (3.2)
to \( r_k = (x_k^T - x_s^T)S(x_k - x_s) + u_k^T R u_k \), to specify a preference for an equilibrium point at \( x_s \).

Figure 4.3(a) shows the total costs when \( x_s^T = \begin{bmatrix} 1 & -1 \end{bmatrix} \). It is clear that the cost reduction after \( \sigma_e \approx 1 \) is much larger for the extended approach. The \( \sigma_e \approx 1 \) is the same scale as \( x_s \) and doing the same experiment for a larger \( x_s \) shows that more exploration is required (not shown).

For one result obtained using \( \sigma_e = 5.6 \), the state values are shown as a function of time in figure 4.3(b). The values for the standard approaches always go to zero while the values for the extended approach go to two different values. The value of \( \dot{\ell} = -0.55 \) brings the equilibrium state closer to \( x_s \). Note that the equilibrium state will not be equal to \( x_s \) because of the costs assigned to the control action that keeps it at \( x_s \).

The simulation experiments have shown that the correct value of \( \ell \) will be found. They have shown that in case of \( w \neq 0 \), the resulting feedbacks \( L_{QL} \) and \( L_{EX} \) will have the same outcome. However, the extended approach requires less exploration to obtain this result. Finally, the experiments showed that the extended approach is also able to learn the set point.

![Figure 4.2. Results of the simulation experiments I and II. The solid lines are the results for the extended approach, the dashed lines the result of the standard approach. The amount of system noise \( \sigma_v = 10^{-4} \) is indicated by the dotted line.](image)
4.5 Simulation Experiments with a Nonlinear System

For a nonlinear system (4.15) the value of $w(x, u, v)$ does not have to be constant. In the previous section we showed that the extended LQRQL approach performs better than the standard approach when a constant $w$ is not zero. If the nonlinearity is smooth then in a small part of the state space the value of $w(x, u, v)$ does not vary too much. If the average value of $w(x, u, v)$ is not zero, the extended LQRQL approach should also perform better. To verify this we did some experiments with a nonlinear system, where we compared the performance of the extended LQRQL approach with the SI and standard LQRQL approach from chapter 3.

4.5.1 The nonlinear system

The mobile robot

As nonlinear system we used a mobile robot system. A description of the robot can be found in appendix B. Further a model is given in appendix B, that describes how the robot changes its position and orientation in the world. The change of orientation does not depend on the position, but the change in position depends on the orientation. This suggests that this can be used as a nonlinear system, where the effect of the orientation on the change of position introduces the nonlinearity.

The task of the robot is to follow a straight line that is defined in the world. The state is given by the distance $\delta$ to that line and the orientation $\alpha$ with respect to that line, as
Figure 4.4. The top view of the robot. The left figure illustrates the general task. The right figure shows our implementation where the line to follow is the positive x-axis. This figure also indicates the robot’s movement, given a positive, negative or zero $u$.

shown in figure 4.4(a). So the state is given by $\mathbf{x}^T = [\alpha \quad \delta]$, where $\delta$ is given in meters and $\alpha$ in radians. Using (B.6) we can express the state transition for the robot given this task:

\[
\begin{align*}
\alpha_{k+1} &= \alpha_k + \omega T, \\
\delta_{k+1} &= \delta_k + \frac{v_t}{\omega} (\cos(\alpha_k) - \cos(\alpha_k + T \omega)).
\end{align*}
\]

The $T$ is the sample time. According to (4.36) the trajectory of the robot describes a part of a circle with radius $\frac{v_t}{\omega}$. There are two control actions: the traversal speed $v_t$ and the rotation speed $\omega$. We gave the traversal speed a fixed value of 0.1 meters per second. As control action $u$ we took the rotation speed $\omega$. Without loss of generality we can take the x-axis of the world as the line to follow. Our simulation setup is shown in figure 4.4(b), where we take $\delta = y$ and $\alpha = \phi$.

We used quadratic direct costs, so

\[
r_k = \mathbf{x}^T \begin{bmatrix} S_\alpha & 0 \\ 0 & S_\delta \end{bmatrix} \mathbf{x} + uRu,
\]

with

\[
S_\alpha = 0.1, \quad S_\delta = 1 \quad \text{and} \quad R = 1.
\]
Comparing the results

Before starting the experiments we will give an indication about the correct feedbacks. We will also show that the value of $\delta$ can be used to vary the value of $w(x, u, v)$. To compare the results of the three approaches we have to test the performance. We will describe how we will test the results.

The task is to find a linear feedback $\hat{L}$ by applying the SI approach and both LQRQL approaches. The extended LQRQL approach also has to find an extra term $\hat{l}$ from the train set. So the resulting feedback for the SI and standard LQRQL approach is:

$$u = \hat{L}x = [\hat{L}_\alpha \; \hat{L}_\delta] \begin{bmatrix} \alpha \\ \delta \end{bmatrix}. \tag{4.39}$$

For the extended LQRQL approach there is an extra term

$$u = \hat{L}x + \hat{l} = [\hat{L}_\alpha \; \hat{L}_\delta] \begin{bmatrix} \alpha \\ \delta \end{bmatrix} + \hat{l}. \tag{4.40}$$

Now we can already determine what the correct feedback should look like. If $\delta > 0$ and $\alpha = 0$, the robot has to turn right. This means that for $u < 0$, $\hat{L}_\delta < 0$ is correct. If $\alpha$ is also positive, the robot has to turn right even more. This means that also $\hat{L}_\alpha < 0$ is correct. The $\hat{l}$ can be used in that state to steer to the right even more, so this should also be negative. However, if $\delta < 0$ the value of $\hat{l}$ should be positive.

The model in (4.36) describes a smooth nonlinear function. This function is symmetric around the origin of the state space, which indicates that the optimal feedback function will go through the origin. So it is possible to use (4.39) as a local linear approximation of the optimal feedback function. When the robot is closely following the line, the value of $w(x, u, v)$ will be very small. The extended LQRQL approach has to result in $\hat{l} = 0$.

For large values of $\delta$, the optimal feedback will move the robot in the direction of the line. This implies that $\alpha$ is no longer very small, so that the nonlinearity in (4.36) will have an effect on the change of $\delta$. The value of $w(x, u, v)$ is no longer zero, so that the (4.39) will not be able to form a good local linear approximation of the optimal feedback function. The extended LQRQL approach can provide a better approximation for a large $\delta$ and should result in $\hat{l} \neq 0$.

In order to compare the results of the different approaches, we need to have a criterion. The relative performance from the previous chapter cannot be used, because we do not know the optimal solution. We used the total costs over a fixed time interval

$$J = \sum_{k=0}^{N} r_k, \tag{4.41}$$

as criterion. The resulting feedback is obtained based on a train set, that was generated by starting at a certain state. To test the feedback we started the robot in the same state. As time interval we took the same number of time steps as used during the generation of the train set. By taking only a short time interval we made sure that we tested the feedback in the same local part of the state-action space where the train set was generated.
4.5.2 Experiment with a nonzero average \( w \).

We first did an experiment with a nonzero average \( w \). We could do this by focusing on a part of the state space where \( \delta \) is large. Under these conditions the extended LQRQL approach should be able to perform better than the SI and standard LQRQL approach.

**The setting**

We took as settings:

- Sample time \( T = 0.35 \) seconds.
- Number of time steps \( N = 57 \). So with \( T = 0.35 \), the robot drives for approximately 20 seconds.
- Gaussian exploration noise with \( \sigma_e = 0.1 \). A higher level of exploration would make it impossible to use similar settings on the real robot, because the tolerated rotation on the real robot is limited.
- Initial feedback \( L = \begin{bmatrix} -10^{-3} \\ -10^{-3} \end{bmatrix} \). This feedback makes the robot move towards the line. On the other hand it is small enough, so that hardly any prior knowledge is included.
- Initial orientation \( \alpha = 0 \).

Because of the exploration we were not able to exactly determine the local part of the state space in which the train set is generated. The robot drives for approximately 20 seconds at at speed of 0.1 meters per second. Thus the robot traverses approximately 2 meters. As initial distance to the line we used \( \delta_0 = 1.5 \) meters, for which the resulting \( \dot{l} \) should be negative. In the worst case the robot stops at \( \delta = -0.5 \), for which \( \dot{l} \) should be positive. On average most of the training samples were obtained in the part of the state space for which \( l \) should be negative, so that we knew that the extended LQRQL had to result in a \( \dot{l} < 0 \). This means that the average \( w \) was not zero. For the SI and standard LQRQL approach \( l = 0 \), so they should perform less than the extended LQRQL approach.

We generated one train set and used the SI, the standard LQRQL and the extended LQRQL approach. The resulting feedbacks of the three approaches were tested by starting the robot in the same initial state. For each test we computed the the total costs according to (4.41). In Table 4.1 the resulting feedbacks and the total costs of the test runs are shown.

<table>
<thead>
<tr>
<th></th>
<th>( \hat{L}_\alpha )</th>
<th>( \hat{L}_\delta )</th>
<th>( \hat{l} )</th>
<th>( J )</th>
</tr>
</thead>
<tbody>
<tr>
<td>SI</td>
<td>-0.2794</td>
<td>-0.7224</td>
<td>×</td>
<td>155.5828</td>
</tr>
<tr>
<td>Standard LQRQL</td>
<td>-0.5861</td>
<td>-0.0330</td>
<td>×</td>
<td>115.3245</td>
</tr>
<tr>
<td>Extended LQRQL</td>
<td>-0.5855</td>
<td>-0.0133</td>
<td>-0.5865</td>
<td>45.8943</td>
</tr>
</tbody>
</table>

**Table 4.1.** Results of the experiment with nonzero average \( w \).
The resulting value of $\dot{I}$ is negative, just as we expected. The values of $\dot{L}$ for both LQRQL approaches are almost the same, but the $\dot{L}$ for the SI approach is different. The total costs of the SI approach is the highest and the extended LQRQL approach has the lowest total costs.

In figure 4.5 five trajectories are shown. The first shows that using the initial feedback without exploration will make the robot move slowly towards the line. The second trajectory shows that the generated train set, primarily depends on the exploration and not on the initial feedback. The other three trajectories are the test runs with the three resulting feedbacks.

The trajectory of the SI approach is the “curly” line in figure 4.5 that moves to the left. Because the value of $\dot{L}_\delta$ is too large, the robot will rotate too much for large values of $\delta$. At the end of the trajectory the value of $\delta$ is so small that the robot no longer rotates around its axis. The trajectory of the standard LQRQL approach resembles that of the initial feedback $L$, although $L$ and $\dot{L}$ are clearly different. The reason is that $\dot{L}_\delta$ is too small, so the robot hardly rotates. Therefore the orientation $a$ will remain very small, so the higher value of $L_a$ does not contribute to the control action. Although the extended LQRQL approach has approximately the same $\dot{L}$, because of $\dot{l}$ it turns faster in the direction of the line. This explains the lower total costs. The nonzero value of $\dot{I}$ suggests that the average $w$ is not zero. Since the extended approach can deal with this situation, it outperforms the other approaches.

![Figure 4.5](image_url). The trajectories in the world. The line with “No Explo” is the trajectory when using the initial feedback $L$. For the line with “Data” the exploration is added to generate the train set. The other three lines are the trajectories of the test runs.
4.5.3 Experiments for different average $w$

The previous experiment has shown that for a large value of $\delta$ the extended LQRQL approach performs best. This is the case for a nonzero average $w$. We also indicated that around the line the average value of $w$ is zero. Here the SI and standard LQRQL approach already have the correct value of $l = 0$. The extended approach has to estimate the correct $\hat{l}$ which can result in an $\hat{l} \neq 0$. This suggests that the extended LQRQL approach will perform slightly less when the train set is generated near the line. We did some experiments to see how the performance of the three approaches depends on the size of the average $w$.

In the same way as in the previous experiment we used the initial distance $\delta_0$ to determine the average $w$. We varied the initial distance $\delta_0$ from 0 to 2 in steps of $\frac{1}{2}$. The estimation results were based on train sets, generated with a random exploration noise. For this reason we generated for each $\delta_0$ 500 train sets, each with a different exploration noise sequence. The SI, standard LQRQL and extended LQRQL were applied to all train sets.

All resulting feedbacks can be tested, but we first looked at the reliability of the outcome. For all three approaches the resulting feedback function depends only on the train set. If the resulting feedback is not good then this is because the train set is not good enough. In chapter 3 we showed that this is the case when the amount of exploration is not enough. If sufficient exploration is used the estimated quadratic Q-function is positive definite. In that case the estimated matrix $H$ only has positive eigenvalues.

We used the same $\sigma_e$ for all train sets. When we have a nonlinear system, we can see $w_k = w(x_k, u_k, v_k)$ as system noise that does not have to be Gaussian or white. Since the amount of exploration must be higher than the amount of system noise, it can turn out that in some train set insufficient exploration was used. Another problem is that the contribution of the feedback function to the control action is very small when the train set is generated close to the line. The trajectory is mainly determined by the exploration. In these situations it can be very hard to evaluate the feedback function, so the estimated Q-function may not be correct. We know for sure that if $H$ has negative eigenvalues, the Q-function is not correct. For both LQRQL approaches we rejected the results with negative eigenvalues for $H$.

The extended LQRQL approach also estimates $\hat{l}$, whose reliability does not depend on the eigenvalues of $H$. If we look at figure 4.1 we see that a small change in the optimal feedback function may lead to a large change in the value of $l$. This means that a small difference in the train set may lead to large difference in the estimated $\hat{l}$. To be able to compare the results with the other two approaches, the situations for which $\hat{l}$ were clearly wrong were also removed. For the extended LQRQL approach we rejected the results for which $|\hat{l}| > 1$. For these values of $\hat{l}$, the $\hat{L}$ hardly contributed to the motion of the robot.

For the SI approach we do not have a criterion that indicates the reliability of the resulting feedback, so we used all train sets. Figure 4.6(a) shows the fractions of the 500 runs that were used for both LQRQL approaches. For the standard LQRQL approach about 75 percent was used. For the extended approach the percentage without negative
4.5. SIMULATION EXPERIMENTS WITH A NONLINEAR SYSTEM

Figure 4.6. The results for different initial $\delta$.

eigenvalues for $\hat{H}$ was already below 50 percent. When also the unreliable $\hat{l}$ results were removed, about one third of the train sets were kept. This indicates that the extended LQRQL approach is the most sensitive to the particular train set.

The resulting feedback functions that were not rejected were tested. In figure 4.6(b) we see the average total costs $J$ for different initial $\delta$ for the three approaches. When figure 4.6(b) is plotted with error bars the total figure becomes unclear, therefore we omitted the errorbars. The maximal standard deviation was 40.8 for $\delta = 2$ for the extended LQRQL approach. The figure clearly shows that the total costs increases as $\delta$ increases, because $\delta$ is part of the direct costs in (4.37). For $\delta_0 = 0$ the SI and the standard LQRQL approach have $J = 0$. This is because the robot is already on the line and facing the right direction. So $x = 0$ and therefore also $u = 0$. Because $\hat{l} \neq 0$ the total costs for the extended LQRQL approach is not zero.

For small values of $\delta_0$, the SI approach has the lowest costs, followed by the standard LQRQL approach. For high values of $\delta_0$ the situation is reversed. The extended approach has the lowest total costs and the SI approach the highest. This is also what we observed in the preliminary experiment.

We can conclude that if the train set is generated close to the line, the average $w$ is almost zero. The SI and standard LQRQL approach perform best. Further away from the line, the average value of $w$ becomes larger. The total costs for the SI approach is much higher than that of the LQRQL approaches. The extended LQRQL approach has the lowest cost as expected.
4.6 Experiments on a Real Nonlinear System

4.6.1 Introduction

We did experiments with a real robot to see whether the simulation results also apply to real systems. The state value was derived from the odometry. This means that the robot keeps track of its position in the world by measuring the speed of both wheels. We then translated this into a value for $\delta$ and $\alpha$.

There are some important differences with the simulated robot:

- The time steps do not have to be constant. State information is not directly available and is obtained via a communication protocol between the two processing boards in the robot. The duration of the communication varies a little. This introduces some noise, because at discrete time steps the state changes are influenced by the duration of the time step.

- The size of the control action that can be applied is bounded. Too high values for $u$ may ruin the engine of the robot. When generating the train set it was possible that actions were tried that could not be tolerated. For safety we introduced the action bounds, such that $u_{\min} = -0.18 \leq u \leq u_{\max} = 0.18$. When a $u$ outside this interval had to be applied to the robot, we replaced it by the value of the closest action bound. This value was also used for the train set. The consequence is that for high values of $\sigma_x$, the exploration noise is no longer Gaussian.

- The robot has a finite acceleration. This is a part of the dynamics of the system was not included in the model used for the simulations. This means that the trajectory during a time step is not exactly a circle. A consequence is that the robot might not respond so quickly on fast varying control action when exploring. So effectively the amount of exploration contributing to the movement of the robot is lower than the amount of exploration that is added to the control action.

- There is wheel spin. The effect of wheel spin is that the robot does not move according to the rotation of the wheels. This can mean that the real position of the robot does not change, while the wheels are rotating. Since we do not use the robot's real position, this does not affect us. But the odometry is based on measuring the speed of the wheels and they accelerate faster during wheel spin. This has the effect that similar control actions can lead to different state transitions.

4.6.2 The experiments

It was infeasible to generate the same amount of data sets as in the simulation experiments. We varied the initial $\delta$ from 0.25 to 1.75 in steps of 0.5. We generated data for four different sequences of exploration noise, so in total we generated 16 data sets. For one exploration sequence the four data sets generated are shown in figure 4.7(a).
4.6. EXPERIMENTS ON A REAL NONLINEAR SYSTEM

In figure 4.7(b) we see the average total costs for each test of the three different methods. We did not remove any train set for the LQRQL approaches. The results in figure 4.7(b) show that on average all three approaches perform almost the same. Only the extended approach has higher total costs for $\delta_0 = 0.25$. The latter agrees with the simulation experiments. If we compare figure 4.7(b) with figure 4.6(b) we see that the performances of both LQRQL approaches are a little less in practice than in simulation. The SI approach seems to perform better in practice than in simulation.

In figure 4.8(a) we see, for all four exploration noise sequences, the total costs as a function of the initial $\delta$ for the SI approach. For one sequence the total costs were very high (for $\delta_0 = 1.75$ the total costs was more than 600.). This is because the resulting feedback function is wrong. It made the robot move away from the line. For the three other sequences we see that the total costs are very low. These low total costs are misleading. What happens in these situations is that the feedbacks found by the SI approach are much too high. This would lead to rotations as in figure 4.5, but the action bounds prevented the robot from rotating. Instead the action applied to the robot alternated between $u_{\min}$ and $u_{\max}$ for the first 15 to 20 time steps. After that the robot was very close to the line and it started following the line as it should. In these cases the low total costs were caused by the action bounds, which indicates that the SI approach did not find the optimal linear feedback!

**Figure 4.7.** The experiment with the real robot.
The actions taken by the resulting feedbacks of both LQRQL approaches were always between $u_{\text{min}}$ and $u_{\text{max}}$. In figure 4.8(b) we see for all four exploration noise sequences the total costs as a function of the initial $\delta$ for the extended LQRQL approach. We noticed that the plot for the standard LQRQL approach looks quite similar. We see that for some exploration sequences the costs are quite low while for some the costs are higher. This indicates that the performance depends on the particular sequence of exploration noise. For one sequence the total costs are as low as for the three sequences of the SI approach. Only the extended approach did not exploit the action bounds. This indicates that the extended LQRQL approach optimized the local linear feedback function for the real nonlinear system.

In the simulation experiments we rejected train sets that resulted in negative eigenvalues for $\hat{H}$ because we considered them unreliable. We did not do this for the experiments on the real robot. For the standard LQRQL approach 9 of the 16 train sets resulted in a positive definite $\hat{H}$ and for the extended LQRQL approach only 5. The total costs for these feedbacks were not always the lowest, because in some cases an “unreliable” feedback performed better. This was caused by the acceleration of the robot. The reliable feedbacks made the robot rotate faster in the direction to the line than the unreliable feedbacks. But due to the acceleration, the robot did not always stop rotating fast enough. Then the robot was facing the wrong direction and had to rotate back. Many unreliable feedbacks turn slower towards the line and are therefore less affected by the acceleration. This implies that the reliability indication based on the eigenvalues of $\hat{H}$ is not appropriate for the real robot if the acceleration is ignored.

Figure 4.8. All 16 performances for SI and extended LQRQL.
4.7 DISCUSSION

In summary, the experiments have shown that the SI approach does not optimize a local linear feedback correctly. Instead it either finds a wrong feedback or it finds feedbacks that are too large, so that most control actions are limited by the action bound. The performances of the standard and extended LQRQL approach are similar to those of the simulation experiments. So we can conclude that they correctly optimize a local linear feedback.

4.7 Discussion

We start this chapter with a presentation of a number of methods for the control of nonlinear systems. Some of these methods were based on local linear approximations of the system. We were interested whether we could use Q-Learning to find a feedback which is locally optimal. We showed that the standard LQRQL approach as presented in chapter 3 would not lead to an optimal linear feedback, and introduced the extended LQRQL approach. In this approach we do not estimate the parameters of a global quadratic function through the origin, but we use the data to estimate the parameters of a more general quadratic function. The consequence for the feedback function is that an extra constant was introduced. In this way the resulting feedback is no longer restricted to a linear function through the origin.

We tested the extended LQRQL approach on a nonlinear system in simulation and on a real nonlinear system. We compared it with the SI and standard LQRQL approach from chapter 3. The results indicate that if we explore in a restricted part of the state space, by sampling for a few time steps, the extended approach performs much better. This means that if a feedback function is based on local linear approximations, the extended approach has to be used. The standard LQRQL should only be used if it is known that the optimal feedback function goes through the origin.

It is possible to use multiple linear models to construct one global nonlinear feedback function. In [44] bump-trees were used to form a locally weighted feedback function. In [17] local linear feedbacks were used for different partitions of the state space of a nonlinear system. The result is equivalent to Gain Scheduling as described in chapter 2. In both cases the resulting local linear feedback functions were obtained by the standard LQRQL approach.

However, for standard LQRQL the local linear models are not only based on the train set but also on the position of the partition with respect to the origin. This means the models are not completely local. A consequence is that the linear feedbacks for partitions far away from the origin will become smaller. This is because the feedback has to go through the origin. The consequence is that it is unlikely that linear models far away from the origin are optimal. This implies that if we want to form a nonlinear feedback based on local linear feedbacks, we have to use the extended approach. The extended LQRQL approach is able to estimate the appropriate set point for each linear model.
4.8 Conclusions

Many control design techniques for nonlinear systems are based on local linear approximations. We studied the use of LQRQL for obtaining a local linear approximation of a nonlinear feedback function. A nonlinear system can be regarded as a linear system with an additional nonlinear correction. If in a local part of the state space the average correction is not zero, the SI and standard LQRQL approach from chapter 3 will approximate the wrong function. We introduced the extended LQRQL approach, that will result in a linear feedback plus an additional constant. The experiments on a nonlinear system have shown that if the additional constant is not zero, the extended approach will perform better.