Chapter 6
Conclusions and Future work

The objective of the research described in this thesis is to give an answer to the question whether Reinforcement Learning (RL) methods are viable methods to obtain controllers for systems with a continuous state and action space. RL is normally applied to systems with a finite number of discrete states and possible actions. Many real systems have continuous state and action spaces, so algorithms for discrete state and action spaces cannot be applied directly.

Some applications of RL to control real systems have been described [65][59][2][61][47]. This indicates that RL can be used as a controller design method. For these applications the training was performed mainly in simulation, which means that the system has to be known. However, one of the characteristics of RL is that it optimizes the controller based on interactions with the system, for which no knowledge about the system is needed. When the training is performed in simulation the potential of RL is not fully exploited.

There are two main reasons for not applying RL directly to optimize the controller of a system. The first reason is that for real control tasks a reliable controller is needed. The RL algorithms used for the control of continuous state space tasks are either based on heuristics or on discrete RL methods. This means that the closed loop performance when RL is applied to a real system is not very well understood. Stability during learning cannot be guaranteed, which explains the hesitation in applying these algorithms directly on real systems.

The second reason is that RL may train very slowly. In simulation the time required to obtain a sufficiently large train set depends on the computing speed. The computation of one state transition can be much faster than the actual sample time on the real system. On a real system, the time required to obtain a sufficiently large train set is determined by the system. The only way to speed this up is to use algorithms that require less training.

In this thesis we looked at Q-Learning, a model-free RL method. We investigated the applicability of Q-Learning for real systems with continuous state and action spaces. We therefore focused on the two issues mentioned above: how can we guarantee convergence in the presence of system noise and how can we learn from a small train set. We developed three methods: LQRQL for linear systems with quadratic costs, Extended LQRQL for local approximations for nonlinear systems and Neural Q-Learning for finding a nonlinear
feedback for nonlinear systems.

### 6.1 LQR and Q-Learning

A well known framework in control theory is the Linear Quadratic Regularization (LQR) task. It is an optimal control task in which the system is linear and the direct costs are given by a quadratic function of the state and action. The objective is to minimize the total future cost. When the parameters of the system are known, the optimal feedback function can be obtained by solving the Discrete Algebraic Riccati Equation. This solution is used to compute the optimal linear feedback.

When the parameters of the linear system are unknown, System Identification (SI) can be used to estimate the parameters. The estimated parameters of the system can be used to compute the optimal feedback. It has been shown [16][44] that the LQR task can be solved by Q-Learning. For this the original Q-Learning algorithm is adapted so that it can deal with the continuous state and action space of the linear system. We called this approach LQR Q-Learning (LQRQL).

It can be proven that the resulting linear feedback will eventually converge to the optimal feedback when sufficient exploration is used. These results only apply when there is no noise. In practice the system noise will always be present, so we investigated the influence of the system noise on the performance of the resulting linear feedbacks. We aimed at determining the amount of exploration required for a guaranteed improvement and compared this with the more traditional SI approach. For a fair comparison between these methods, we used a batch least squares estimation. So unlike the recursive least squares approach in [16], our results only depend on the train sets used.

To show the influence of the system noise on the performance we introduced the exploration characteristic, in which four types of outcomes can be distinguished for the SI and LQRQL approach. If the amount of exploration is:

I Much too low: No feedback can be computed.

II Too low: The resulting feedback will be the same as that used for the generation of the train set.

III Almost sufficient: The resulting feedback can be anything, depending on the sequence of system noise and exploration.

IV Sufficient: The parameters of the Q-function and system approach the correct value. The new feedback will be an improvement.

Only the type IV outcome is useful, so sufficient exploration is required. For this we derived that the SI approach requires at least more exploration than there is noise in the system. Furthermore we derived that the LQRQL approach requires more exploration than the SI approach.
6.1.1 Future work

**Eigenvalues:** The Q-function is given by a sum of positive definite quadratic functions (the reinforcements), so the correct Q-function is a positive definite quadratic function as well. This can be verified by looking at the eigenvalues of the matrix that represents the parameters of the Q-function. Negative eigenvalue are only possible if the parameters of the Q-function are not estimated correctly. This can only be the case for the type II and III outcome. So negative eigenvalues imply that insufficient exploration was used to generate the train set. In our experiments we always found negative eigenvalues for insufficient exploration, but we cannot guarantee that the estimated Q-function will never be positive definite for insufficient exploration. In order to use the eigenvalues as reliability measure, it still has to be proven that insufficient exploration will never result in a positive definite Q-function.

**Exploration:** The LQRQL approach requires more exploration than the SI approach. For real systems adding disturbance to the control action is not desirable, so to enhance practical applicability the required amount of exploration has to be reduced. One simple way of doing this is by using the SARSA variant of Q-Learning. Q-Learning uses the next action according to the feedback to compute the temporal difference. This does not include the exploration. In SARSA the action taken at the next time step is used and this also includes the exploration at that time step. So each exploration added to the control action is used twice. Experiments have shown that this approach only requires the same amount of exploration as the SI approach. However, SARSA introduces a bias in the estimation, making it perform worse for very high amounts of exploration. These results are only experimental and should be verified theoretically.

6.2 Extended LQRQL

LQRQL was developed for linear systems and not for nonlinear systems. Many control design approaches for nonlinear system are based on local linear approximations of the nonlinear system. LQRQL can be used to obtain a local linear approximation of the optimal feedback function. To investigate the consequences of the nonlinearity it is possible to write the model of the nonlinear system as a linear system with a nonlinear correction.

The linear feedback of the SI and LQRQL approach are not always appropriate feedback functions to approximate the optimal feedback function. An additional offset can be included in the feedback function. This will allow for a better local approximation of the optimal feedback function in case the average nonlinear correction is large in the region of interest. To obtain the value of the offset, the parameters of a more general quadratic Q-function have to be estimated. We showed that these parameters can be estimated in the same way as the original standard LQRQL approach. We called this new approach the extended LQRQL approach. Our experiments on a simulated and on a real nonlinear system confirmed that the extended LQRQL approach results in a better local approxi-
approximation of the optimal feedback function. We can conclude that the extended LQRQL approach has to be applied if a nonlinear feedback function is based on multiple local linear approximations.

6.2.1 Future work

On-line learning: Throughout this thesis we used batch learning, which means that learning starts when the complete train set is available. We did this to make fair comparisons between the different methods possible. Usually RL is used as an on line learning method. The recursive least squares approach for the LQR task in [16][44] is an on-line approach. Although our Q-function is quadratic, it can be seen as a function approximator that is linear in the parameters. For these function approximators the conditions are known for which convergence is guaranteed when on-line TD(λ) learning is used [29][31]. How the on-line TD(λ) approach compares with the recursive least squares approach is not known and should be investigated.

6.3 Neural Q-Learning

Function approximators have been used in RL. One approach uses the actor/critic configuration. One function called the critic is used to approximate the Q-function. The other function is called the actor and is used to approximate the feedback function. Both approximators have to be trained, where the approximation of the actor is trained based on the critic. This makes the training procedure rather tedious and the outcome is hard to analyze.

The LQRQL approach can be combined with a feed-forward neural network approximation of the Q-function. In this case there is no actor because in LQRQL the linear feedback function follows directly from the parameters of the Q-function. So only one function approximator has to be trained. We called this approach Neural Q-Learning.

To obtain a nonlinear feedback function for a nonlinear system using Neural Q-Learning, first a linear feedback has to be determined. The derivatives with respect to the inputs of the network give the parameters that would be estimated by the LQRQL approach. These parameters depend on the state and control action, and therefore it is not possible to directly derive a nonlinear feedback function. It is possible to ignore the tangent hyperbolic activation functions of the hidden units to obtain a globally valid linear feedback function. The linear feedback function can be used to compute the control action that is necessary to determine the nonlinear feedback function. The resulting nonlinear feedback function can be regarded as a locally weighted function, where each hidden unit results in a local linear feedback. The state value determined the weighting of these local linear function.

Experiments were performed on a simulated and on a real nonlinear system, where the goal of the experiments was to obtain global nonlinear feedback functions. We compared Neural Q-Learning with Gain Scheduling. Gain Scheduling can be performed by making for local partitions in the state space a local approximation using the extended LQRQL
approach. The experiments have shown that the neural Q-learning approach requires less training data than the Gain Scheduling approach. Therefore Neural Q-Learning is better suited for real systems.

6.3.1 Future Work

Value Iteration: The neural Q-Learning approach we presented, is still based on policy iteration. The original Q-Learning approach is based on value iteration. Because the feedback function directly follows from the approximated Q-function it is now also possible to apply value iteration. This then can be combined with an online learning approach. Whether Neural Q-Learning using Value Iteration is better than Policy Iteration still should be investigated.

Regularization: The weights of the hidden units should not be too large. In supervised learning there exist approaches for regularization. One way is to assign costs to the size of the weights. The consequences of this modification of the learning for Neural Q-Learning are unknown. It could have the effect that the resulting nonlinear feedback function no longer performs well. This means that further research is necessary to determine whether regularization can be applied.

6.4 General Conclusion

Reinforcement Learning can be used to optimize controllers for real systems with continuous state and action spaces. To exploit the benefits of reinforcement learning the approach should be based on Q-Learning, where the feedback function directly follows from the approximated Q-function. The Q-function should be represented by a function approximator, of which the parameters are estimated from the train set. If it is known that the system is linear, then the parameters of a quadratic Q-function can be estimated. This is the LQRQL approach. In case there is no knowledge about the system, the more general Neural Q-Learning approach can be applied. This will give a globally valid linear feedback function with a local nonlinear correction.