Learning (near) Time Optimal Control for Flexible Robot Joints

Martin Riedmiller
Machine Learning Lab
University of Freiburg, Germany
Email: riedmiller@informatik.uni-freiburg.de

Roland Hafner
Cognit GmbH
Balgheim, Germany
Email: rhafner@cognit-labs.de

***Abstract*—This short paper describes the application of a model free, learning neural controller, that is able to optimally control a flexible joint of a robot arm.**

I. Introduction

Robotic arms with series elastic actuators [3] provide mechanical compliance in the joint actuation that can help to simplify many challenging real world tasks like force control in constrained situations or movements with high dynamic and speed [6]. The decoupling of the joints and the motor gearboxes by using some kind of torsion or linear springs results in a low-pass filtering of force and torque peaks that helps to provide an increased safety of operation, for example for the human interaction with the robot arm. On the other hand the introduction of elastic elements in the joint actuation results in a reduced torque bandwidth and the demand for damping the oscillations that increases the controller complexity, especially in the context of time optimal control.

Our work tackles the set-point feedback and trajectory control of a single compliant joint. Instead of designing a model based controller by hand we use a Reinforcement Learning approach to learn the controller by pure interaction with the system. The applied controller learning scheme is based on a neural batch RL approach, the Neural Fitted Q-Iteration (NFQ) [4] and its extension to continuous action spaces (NFQCA, see [2, 1]). This approach proved to be successful in learning high quality control policies for different challenging nonlinear control tasks [2].

In this contribution we want to present first results of our recent work, where we applied the learning RL controller to time optimal feedback control of a simulation model of a single elastic joint and link combination. In contrast to earlier publications we show also that is possible to learn a trajectory tracking control for this application. Especially we want to show that the amount of required interaction time and the quality of the learned control law, the robustness and the generalization, fits the requirements to apply the learning controller to the real system.

A. Simulation Model of a Single Joint

The used simulation model is a combination of a DC motor, that is coupled to a link with length l and mass m by a torsion spring. Figure 1 shows the schematic setup. We assume that the joint position, θ_j, and the motor position, θ_m, can be measured independently. This is realized in most elastic joint setups by using independent encoders on the motor and link side. Along these values the motor current, c_m, the angular velocity of the joint, $\dot{\theta_j}$, and the velocity of the motor, $\dot{\theta_m}$, are available by measurement or numeric filtering.

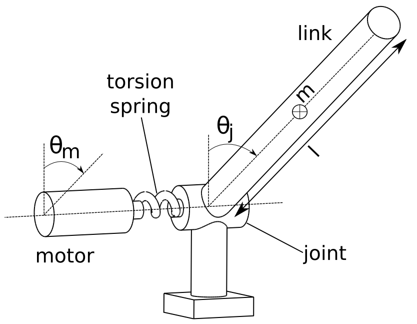

Fig. 1: The model of a single flexible link under gravity.

A simplified dynamic model of the setup in Figure 1 can be formulated as a dynamic system with state vector, $X = (\theta_m, \dot{\theta_m}, c_m, \theta_j, \dot{\theta_j}) = (x_1, x_2, x_3, x_4, x_5)$, and the dynamic equations 1 - 5.

$$\begin{aligned}
\dot{x_1} &= \dot{\theta_m} && (1)\\
\dot{x_2} &= \frac{1}{J_m}(k_t c_m - c(\theta_m - \theta_j)) && (2)\\
\dot{x_3} &= \frac{1}{L_m}(R_m c_m - k_e \dot{\theta_m} + u) && (3)\\
\dot{x_4} &= \dot{\theta_j} && (4)\\
\dot{x_5} &= \frac{1}{J_l}(c * (\theta_m - \theta_j) && (5)\\
&\quad -k_f * (\dot{\theta_m} - \dot{\theta_j}) + 0.5 lmg \sin(\theta_j))
\end{aligned}$$

Which is a combination of a DC motor with inertia J_m (inclusive gearbox), torque constant k_t, back emf constant k_e, electrical resistance R_m and inductance L_m, a torsion spring with spring constant c and viscous friction factor k_f, and a link with length l, inertia J_l and mass m.

A controller for this setup has to influence the joint position θ_j according to a given set-point or a set-point trajectory by setting adequate voltage u to the motor. As the motor is coupled to the link by a torsion spring, the controller has to compensate not only for the gravity in each position, but also for the spring torques.

II. Learning Time Optimal Set-Point Control

In time optimal set-point control the dynamic system should be controlled so that the variable under control, θ_j, reaches a given set-point, θ_j^d, in a minimum of time steps, and retains it. A deviation of the variable under control and the set-point can occur due to disturbances on the system or the change of the set-point and should be canceled by the control law, given the physical constraints of the system (e.g. a maximal voltage).

With NFQCA (see [2] and [1]), we use a model free batch Reinforcement Learning scheme, that is based on Q-learning [5], in combination with neural networks. When used in a growing batch configuration, this approach allows to learn high quality policies from scratch, with almost no a priori knowledge, just by interaction, for a broad range of applications and sequential decision problems. For learning time optimal set-point control, we use this approach with a very general time optimal and precise formulation of the direct cost signal, that is described in [2].

In the Reinforcement Learning setup the state of the MDP is a combination of the full state of the dynamic system under control and the set-point. In our context we defined the state of the MDP as $s = (\theta_m, \dot{\theta_m}, c_m, \theta_j, \dot{\theta_j}, e = \theta_j^d - \theta_j)$. For simplicity the action a of the MDP is defined in the interval $[-1, 1]$, where the applied voltage is defined as $u = a * vcc$, with vcc is the maximal available voltage.

The learning process starts with a random policy and interacts with the system in sequences of 150 time steps length (control interval is 2 milliseconds). Following the principles of the growing batch scheme after each sequence a new neural policy is computed with NFQCA. The learning process required 92 iterations (interaction sequences and learning updates) to compute a high quality control law for this application. This corresponds to less than 30 seconds of pure interaction time with the system that was used for training the controller.

For comparison of the control performance of the learned controller, we use a hand designed reference controller, that we designed for a real robot application. The reference controller is a combination of a PID controller with state based compensation of gravity and spring torques, that was tuned for the simulated link. Figure 2 shows a control trajectory of the fully tuned reference controller. As can be seen in comparison to Figure 3, the learned controller can accelerate the system much faster and is able to damp it, to reach the set-point in fewer time-steps than the reference controller.

To quantify the performance of a controller we apply it to 50 randomly chosen set-points. To benchmark the controller, we report the average number of time steps, $\bar{N}$, the controller needs to reach the given set-points, the average steady-state error $\bar{e^{\infty}}$, and the maximal steady-state error, e^{∞}_{max}, over the given set-points (see [2] for a precise definition of the performance citeria.)

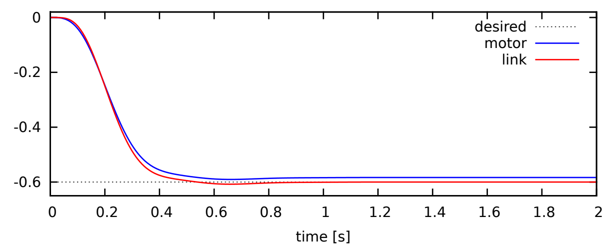

Fig. 2: Trajectory of the manually tuned reference controller.

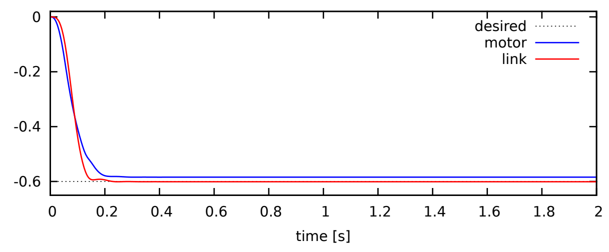

Fig. 3: Trajectory of the learned controller showing significantly improved acceleration and damping.

The results of benchmarking the learned controller against the reference controller in table I, shows that the learned controller is much faster and more precise than the reference controller.

	$\bar{N}$	$\bar{e^{\infty}}$	e^{∞}_{max}
reference controller	135.40	0.015	0.028
learned controller	40.88	0.003	0.006

TABLE I: Results of a controller benchmark.

1) Controller Robustness: As an example we show the robustness of the controller against variations of the link mass. The learning controller was trained on a system with a link mass of $m = 1kg$, to which also the reference controller was tuned.

	learned controller			reference controller		
m	$\bar{N}$	$\bar{e^{\infty}}$	e^{∞}_{max}	$\bar{N}$	$\bar{e^{\infty}}$	e^{∞}_{max}
0.5	44.56	0.002	0.004	136.50	0.017	0.031
1.0	40.88	0.003	0.006	135.40	0.015	0.028
1.5	41.56	0.003	0.007	154.84	0.020	0.037
2.0	52.58	0.007	0.010	682.34	0.143	0.214
2.5	63.02	0.011	0.019	x	x	x
3.0	81.28	0.014	0.026	x	x	x

TABLE II: Controller benchmark with varying mass of link. Both controllers were trained/tuned for a system of $m = 1kg$.

Table II shows the results of the benchmarking of the controller performance under variation of the link mass. The classical reference controller has a limited operating range and gets completely unstable at 2.5kg. In contrast the learned controller shows a remarkable robustness, that we observed in many other applications. To further enhance the robustness of the controller against varying parameters, one way to go is to include the variation of the parameters already during the training phase (see [1] for details).

III. Learning Trajectory Control

For most real world application, the set-point control of the system is not sufficient and has to be replaced by a trajectory control scheme. In our setup, for the trajectory control, a desired trajectory is given by θ_j^d and $\dot{\theta}_j^d$. For a first demonstration we use sinusoidal set-point trajectories. The learned set-point controller of section II will fail to follow precisely a continuously changing set-point trajectory, as the controller learned to reach a given set-point only in steady state condition (see Figure 4). To formulate the

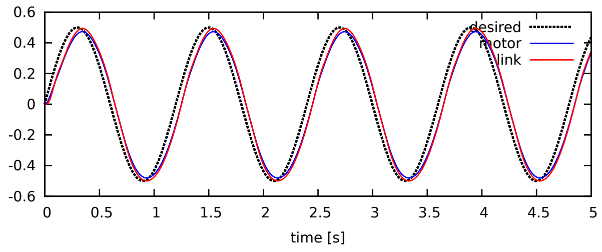

Fig. 4: The learned controller of section II in trajectory control.

complete trajectory control problem, the state of the MDP has to be extended, to reflect the additional source of dynamic in the MDP. In our case, we extended the MDP state to $s = (\theta_m, \dot{\theta_m}, c_m, \theta_j, \dot{\theta_j}, \dot{\theta_j^d}, e = \theta_j^d - \theta_j)$. For the learning procedure we used only two sinusoidal trajectories, with the same amplitude (0.5 rad), that have periods of 1.6s and 0.8s, around the rest position. The length of the trajectories were set to 300 time steps and the control interval set to 5 milliseconds. All other parameters of the learning process (including neural network sizes and topologies) are the same as for the last section.

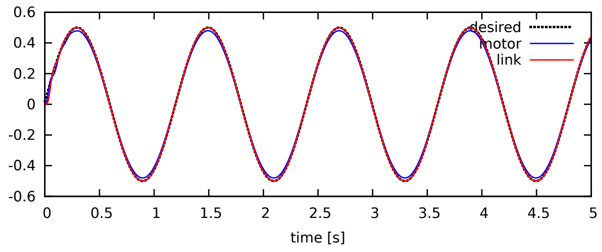

Fig. 5: The learned trajectory controller.

After only 120 sequences (3 minutes of pure interaction time) the new controller produced a high quality control law for the two given trajectories. In Figure 5 an example trajectory is given for the learned controller. The mean absolute trajectory error over the whole training trajectories (over 1000 time steps) is $\bar{e^\infty} = 0.002$ ($e_{max}^\infty = 0.003$).

To test the generalization capabilities of the learned controller, we run 50 sequences with amplitudes in the range of $[0.2, 0.6]$ and periods in the range $[0.6, 2.0]$, with the learned controller. On this test set, the controller showed a mean absolute trajectory error of $\bar{e^\infty} = 0.004$ ($e_{max}^\infty = 0.007$). This shows, that the controller learned a good control law for a wide range of trajectories, only from two simple training trajectories.

IV. Conclusion

We presented recent results of learning time-optimal and trajectory control of a flexible robot joint with a neural batch Reinforcement Learning approach. The controller learns purely by interaction with the system and has no prior knowledge of the system dynamics or system parameters. As we showed for a simulated joint, the learned control laws are of high quality, generalize well over a broad working area and can be learned in reasonable time. Also a good robustness of the learned controller was observed.

The main idea for application of this controller is to learn it directly on a robot arm, built of a series of links. Ongoing research will show if it is possible to find representative configurations, dynamic situations and trajectories to learn such a robust controller for each individual link in reasonable time. The results obtained so far a highly encouraging.

V. Acknowledgements

Parts of this work have been developped in conjunction with a research cooperation between HARTING KGaA, Espelkamp and Cognit GmbH, Balgheim. The authors therefore wish to thank HARTING KGaA for giving permission to publish this work.

References

[1] Roland Hafner. *Dateneffiziente selbstlernende neuronale Regler*. PhD thesis, Univerity of Osnabrueck, 11 2009.

[2] Roland Hafner and Martin Riedmiller. Reinforcement learning in feedback control. *Machine Learning*, pages 1–33, 2011. ISSN 0885-6125. URL http://dx.doi.org/10.1007/s10994-011-5235-x.

[3] G. A. Pratt and M. M. Williamson. Series elastic actuators. *iros*, 01, 1995. URL http://dx.doi.org/10.1109/IROS.1995.525827.

[4] Martin Riedmiller. Neural fitted q iteration - first experiences with a data efficient neural reinforcement learning method. In *Proc. of the European Conference on Machine Learning, ECML 2005*, Porto, Portugal, October 2005.

[5] Christopher J. Watkins and Peter Dayan. Q-learning. *Machine Learning*, 8(3):279–292, May 1992.

[6] S. Wolf and G. Hirzinger. A new variable stiffness design: Matching requirements of the next robot generation. In *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, pages 1741–1746. IEEE, 2008.