Differences and similarities between reinforcement learning and the classical optimal control framework

Simon Gottschalk¹,* and Michael Burger¹,**

¹ Fraunhofer ITWM, Fraunhofer-Platz.1, 67633 Kaiserslautern

In this contribution, we discuss Reinforcement Learning as an alternative way to solve optimal control problems. Especially for biomechanical models, well-established classical techniques can become complex and time-consuming, because biomechanical models have often much more actuators than degrees of freedom. Furthermore, the solution of such a technique is normally only applicable to this specific setting. This means, that a slightly change of the initial value of the model or the desired end position does make the computed solution useless. We give a short overview to Reinforcement Learning and apply it to an optimal control problem containing the above mentioned challenges. We use an algorithm, which updates the weights and biases of a neural network, which takes the role of a controller, using simulated trajectories of the model generated by the current neural network.

1 Optimal control techniques

We focus on the usability of Reinforcement Learning (RL) [1] in order to find the optimal control of biomechanical models performing desired tasks. Typically, such an optimal control problem consists of equations of motion representing the biomechanical model and forming the constraints of the optimization problem. This ensures, that the motion stays consistent to the model. Additionally, an objective function is given and one is typically interested in minimizing (resp. maximizing) this function with respect to the control. In order to solve such a problem, there are several well established ways.

RL benefits from another representation of this optimal control problem. It is assumed that instead of the discrete equations of motion only a transition probability is known. That means, only a distribution that describes the transition from the state \( x_t \in S \), where \( S \) is the state space, at time \( t \) to the state \( x_{t+1} \in S \) at time \( t+1 \) is needed. Here, a Markov property for the model is assumed and thus, the transition distribution from time \( t \) to time \( t+1 \) only depends on the state \( x_t \in S \) and a given control \( (\text{or action}) \ u_t \in A \) from the control space \( A \) and not on previous states and controls. This transition probability is denoted as \( P_{u_t}(x_t, x_{t+1}) \). This is the only restriction RL needs with respect to the dynamical model, which is a huge advantage in comparison to the classical framework, where we need to have the equations of motion. This allows, for instance, to apply RL to real world models, where the equations are not fully known or disturbances can not be modelled. The only missing part in the RL-framework is the objective function. It is necessary to define the desired task. For this, a reward function \( r : S \times A \rightarrow \mathbb{R} \) is defined, which rates the current situation with respect to the task. Overall, the quadruple \( (S, A, P_{u_t}, r) \) forms a Markov decision process [2].

The RL algorithms aim to maximize the expected value of the summed up rewards of one simulation trajectory. There are several ways to do this. First, one has to distinguish between policy-based, which directly trains a controller (also policy) and value function-based algorithms improving step by step a so called value function, which can be used to evaluate the next action. Our application trials convinces us to use a policy-based algorithm, where we parametrize the policy by a neural network. To be more precise, the policy \( \pi(\theta|x_t) \), which gives us the probability for the next control \( u_t \) for a known current state \( x_t \), consists of a neural network getting the \( x_t \) as input and outputs the parameters of a Gaussian distribution from which we can sample the next control. The weights and biases of the neural network (represented by \( \theta \)) can now improved iteratively in order to get a better controller. These improvements result from iterative methods for the following RL optimization problem:

\[
\max_{\theta} \mathbb{E} \left[ \sum_{(x_t, u_t) \in \tau} r(x_t, u_t) \right],
\]

where \( \tau \) denotes the set of state-control pairs of the trajectory. Conceivable algorithms in order to solve this optimization problem are, among other, gradient based methods [3] or trust region methods [4].

2 Human arm

At this point, we consider a biomechanical model, which was considered in [5]. The multibody model of a human arm consists of rigid bodies representing the bones and 29 Hill’s muscle models in order to actuate the arm (see Fig 1b). The goal
of our motion is to reach a target with the finger tip of the arm. We train one controller for all points on a one eighth circle (represented by ten points in Fig. 1b). The neural network gets the state of the model as input as well as the desired end position of this trajectory. In the classical optimal control framework, one would have solved each target for itself. This is, besides the advantages of RL to be able to handle overactuated systems (system with much more actuators than degrees of freedom), another useful property of RL. We applied a Proximal policy optimization technique [6] based on the implementation by P. Coady [7]. The reward rates every state-action pair, except the last state of a trajectory, with a $-0.1$. The reward for the last one is ten times minus the distance between the finger tip and the desired end position. If the finger tip does reach the end position a positive reward is outputted. The training takes a few days on a standard laptop computer (Intel® Core™ i7).

Fig.1b, Fig.1d and Fig.1c show snapshots of a trajectory reaching a point. The Fig.1a shows 45 trajectories of the finger tip for random targets on the eighth circle. The summed up rewards during the training in each iteration is plotted in Fig.1e.

### 3 Conclusion

We have seen that RL is suitable for solving optimal control problems with more actuators than degrees of freedom like the model of a human arm. A policy can be trained, which can also handle small changes in the desired end position. Thus, RL is able to handle the challenges given by the biomechanical model and its task. The generated movement looks human-like and plausible.

**Acknowledgements** The authors are grateful for the funding by the Federal Ministry of Education and Research of Germany (BMBF), project number 05M16UKD.

### References

[1] R. S. Sutton and A. G. Barto, Introduction to Reinforcement Learning, 1st edition (MIT Press, Cambridge, MA, USA, 1998).
[2] M. L. Puterman, Markov Decision Processes: Discrete Stochastic Dynamic Programming, 1st edition (John Wiley & Sons, Inc., New York, USA, 1994).
[3] R. J. Williams, Machine Learning 8(3), 229–256 (1992).
[4] J. Schulman, S. Levine, P. Abbeel, M. I. Jordan, and P. Moritz, Trust region policy optimization, in: ICML, JMLR Workshop and Conference Proceedings, Vol. 37 (JMLR.org, Lille, France, 2015), pp. 1889–1897.
[5] M. Roller, S. Björkenstam, J. Linn, and S. Leyendecker, Optimal control of a biomechanical multibody model for the dynamic simulation of working tasks, Proceedings of the 8th ECCOMAS Thematic Conference on Multibody Dynamics (Prague, 2017), pp. 817–826.
[6] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, CoRR abs/1707.06347 (2017).
[7] P. Coady, Ai gym workout, 2017, https://learningai.io/projects/2017/07/28/ai-gym-workout.html, 2018-10-26.