Solving the swing-up and balance task for the Acrobot and Pendubot with SAC

Chi Zhang[1]
[1] Robotics Innovation Center, DFKI GmbH, Bremen, Germany

Motivation

Robotic control is a complex problem and is approached in many different ways, such as classical motion planning, optimal control methods, and learning-based methods. Extensive research has been conducted on complex robotic motion control, with gaining widespread adoption in reinforcement learning methods recent times.
Reinforcement learning [1] is a machine learning approach where an agent learns to make decisions by interacting with an environment and receiving feedback in the form of rewards or penalties. The idea of reinforcement learning shares a common point with optimal control, which is a computational approach that seeks to determine the inputs to a system over time in order to optimize a defined performance measure, accounting for system dynamics and constraints.
Reinforcement learning plus optimal control has a potential to leverage both learning from data and mathematical optimization to achieve optimal performance in complex and dynamic environments, thereby enabling efficient decision-making and action.

Problem setup

Double pendulum [2] is the perfect setup for testing newly designed control algorithms for its highly nonlinear and chaotic behavior. There are two underactuated variations of the double pendulum setup. If it is actuated on the shoulder joint, it is called "pendubot", on the contrary, if it is actuated on the elbow joint, it is called "acrobot".

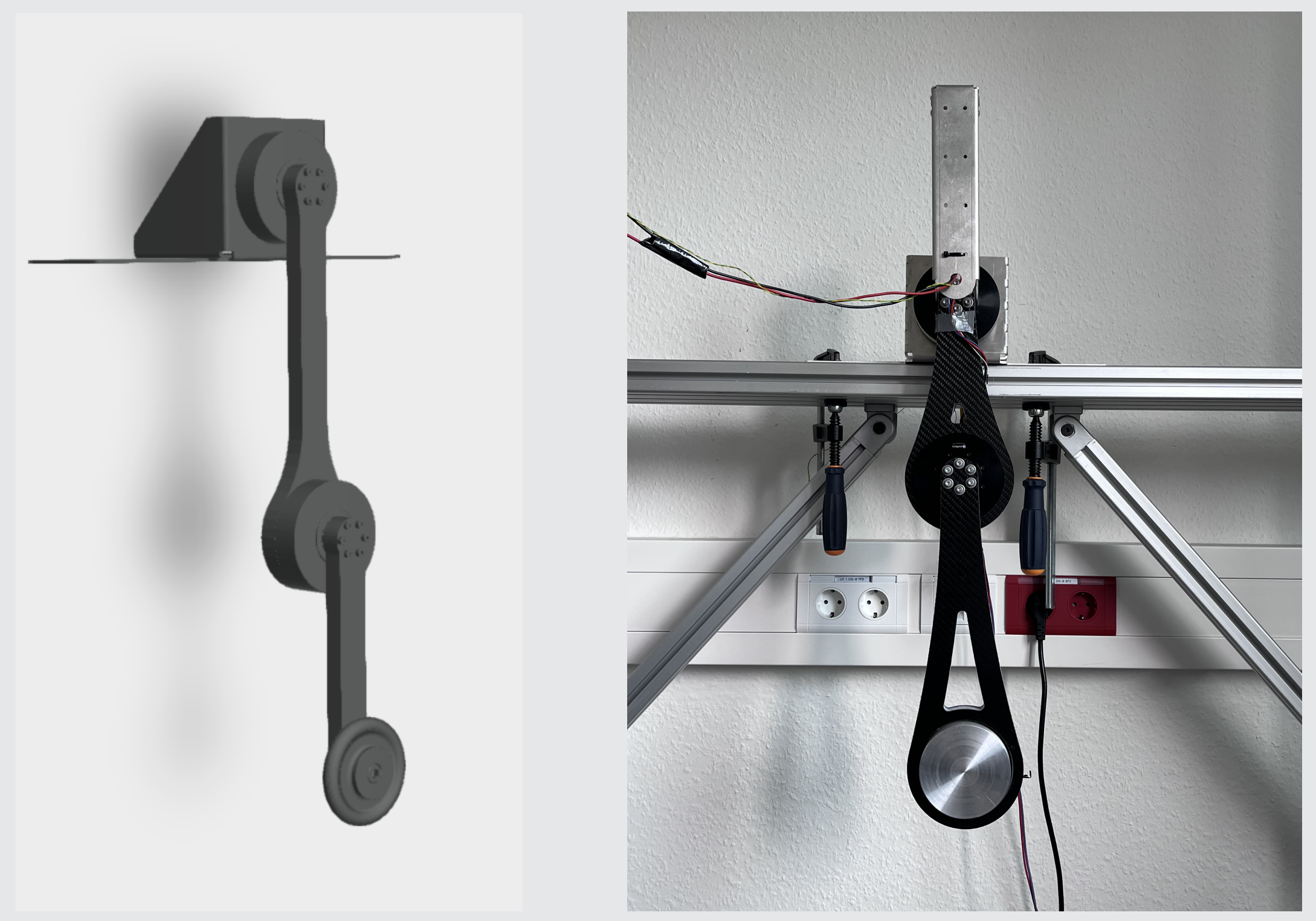


Figure: Double pendulum setup in CAD and in real world

The objective revolves around accomplishing two primary tasks: swinging the double pendulum from its lowest position to its highest point, and maintaining stability at the highest point.

Methodology: SAC+LQR

To achieve the swing-up task,we employ the classical model-free reinforcement learning algorithm known as Soft Actor-Critic (SAC) to train a policy which is able to reach the region of attraction (RoA) of a continuous time linear quadratic regulator (LQR) controller. As soon as the system enters the RoA, we transition to the LQR controller to stabilize the entire system.
Soft Actor Critic (SAC) [3] is a widely used reinforcement learning algorithm tailored for continuous action spaces. SAC uses an actor-critic structure, where the actor selects actions based on the policy, and the critic evaluates state-action pairs. Entropy regularization is a key aspect of SAC, promoting exploration and enhancing learning by maximizing action randomness.
LQR [4] excels at controlling nonlinear systems near a linearized operating point. Theoretically, we can approximate the RoA [5] in the 4D state space with an ellipsoid. Optimizing the RoA volume using numerical methods, we achieve the largest possible RoA volume by tuning hyperparameters of the LQR controller.

Reward shaping

We've devised a 3-stage reward approach. The first stage utilizes a quadratic reward function to promote smooth swinging within a few training sessions. Upon reaching a threshold line with the end effector, a second level of reward r_{line} is introduced. The third reward level, r_{LQR}, encourages the agent to stay within the Region of Attraction (ROA) of the LQR controller. This encourages a seamless transition from the SAC controller to the LQR controller.To discourage the agent from exploiting rewards by spinning at excessive speeds, a significant penalty $-r_{vel}$ is implemented for any speed exceeding v_{thresh}.
The full equation for this reward function is:

$$
\begin{aligned}
r(x,u) = &-(x-x_g)^T Q_{train}(x-x_g) - u^T R_{train} u \\
&+ \begin{cases} r_{line} & \text{if } h(p_1,p_2) \geq h_{line} \ , \\ 0 & \text{else} \end{cases} \\
&+ \begin{cases} r_{LQR} & \text{if } (x-x_g)^T S_{LQR}(x-x_g) \geq \rho \ , \\ 0 & \text{else} \end{cases} \\
&- \begin{cases} r_{vel} & \text{if } |v_1| \geq v_{thresh} \ , \\ 0 & \text{else} \end{cases} \\
&- \begin{cases} r_{vel} & \text{if } |v_2| \geq v_{thresh} \ , \\ 0 & \text{else} \end{cases}
\end{aligned}
$$

where x is the current state, x_g is the goal state, u is the control input, and the end effector height $h(p_1, p_2)$ is given by: $h(p_1,p_2) = -l_1\cos(p_1) - l_2\cos(p_1+p_2)$

Simulation result

As shown here, we have achieved successful results in a simulation environment.

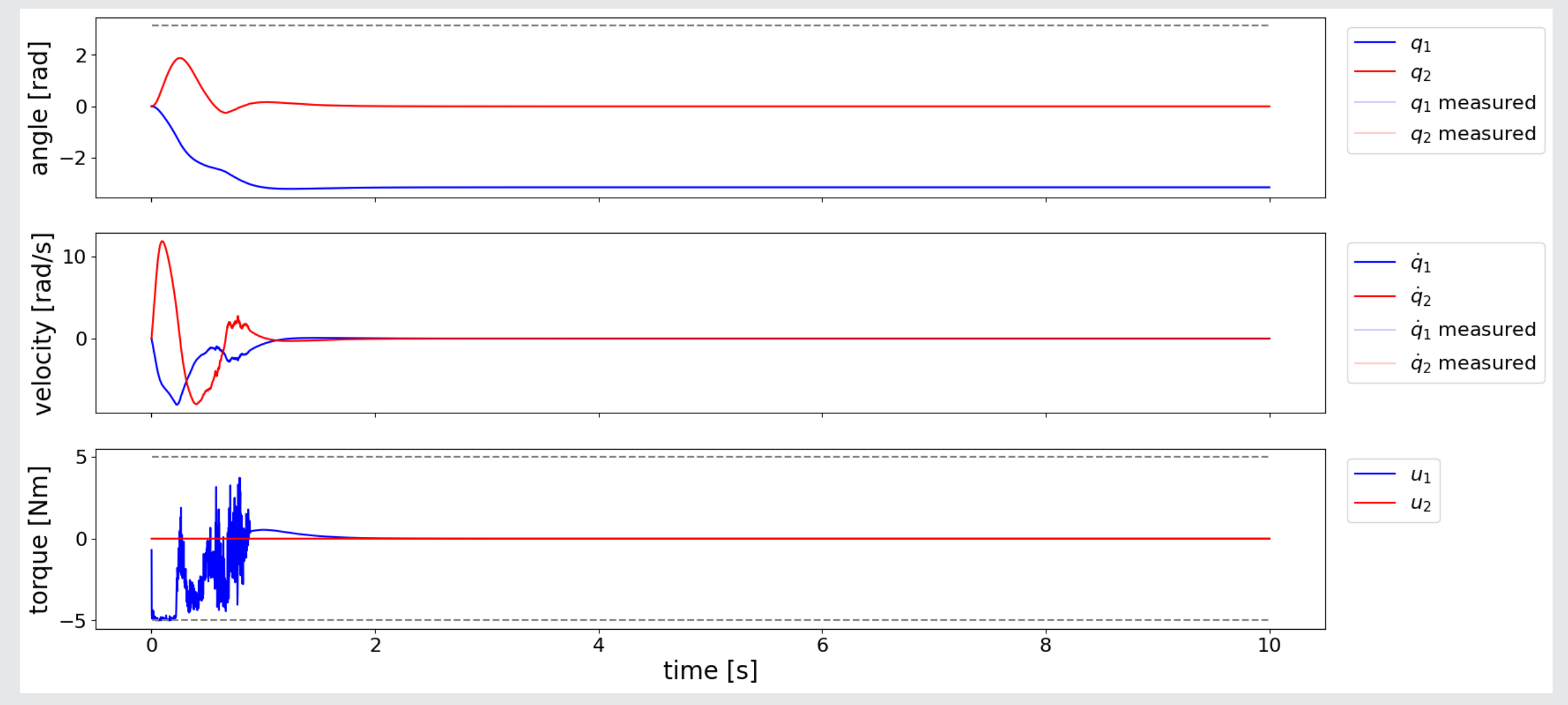


Figure: Successful pendubot simulation

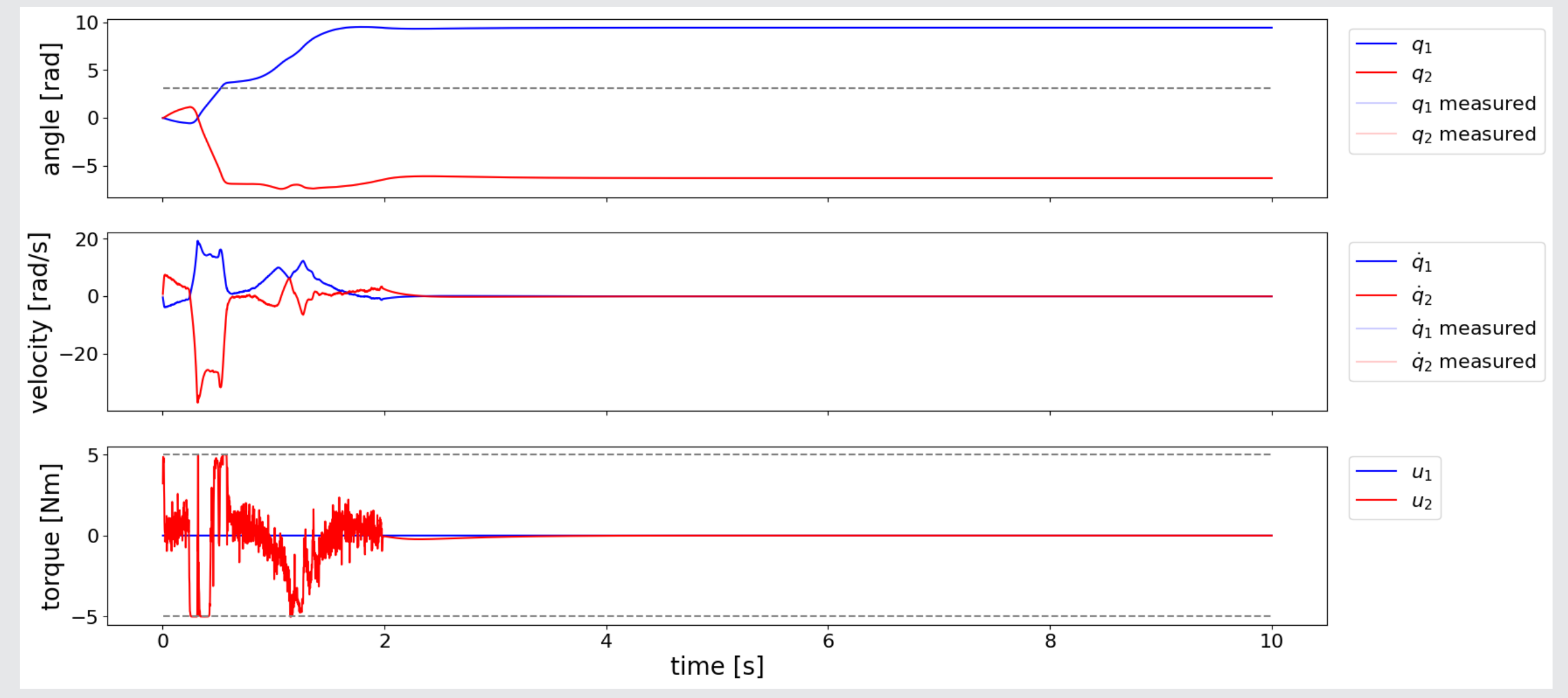


Figure: Successful acrobot simulation

Pendubot swings up in one second, acrobot in about two. Both stabilize near unstable points for an extended period. According to simulation leaderboard, our model delivers fair results in terms of maximum torque, integrated torque, and torque cost. However, we have made some compromises in terms of torque smoothness and energy efficiency.

Sim2Real gap

When transitioning the successful simulated model to the real system, we encounter issues such as control signal latency, torque tracking, model uncertainty, and other disturbances.

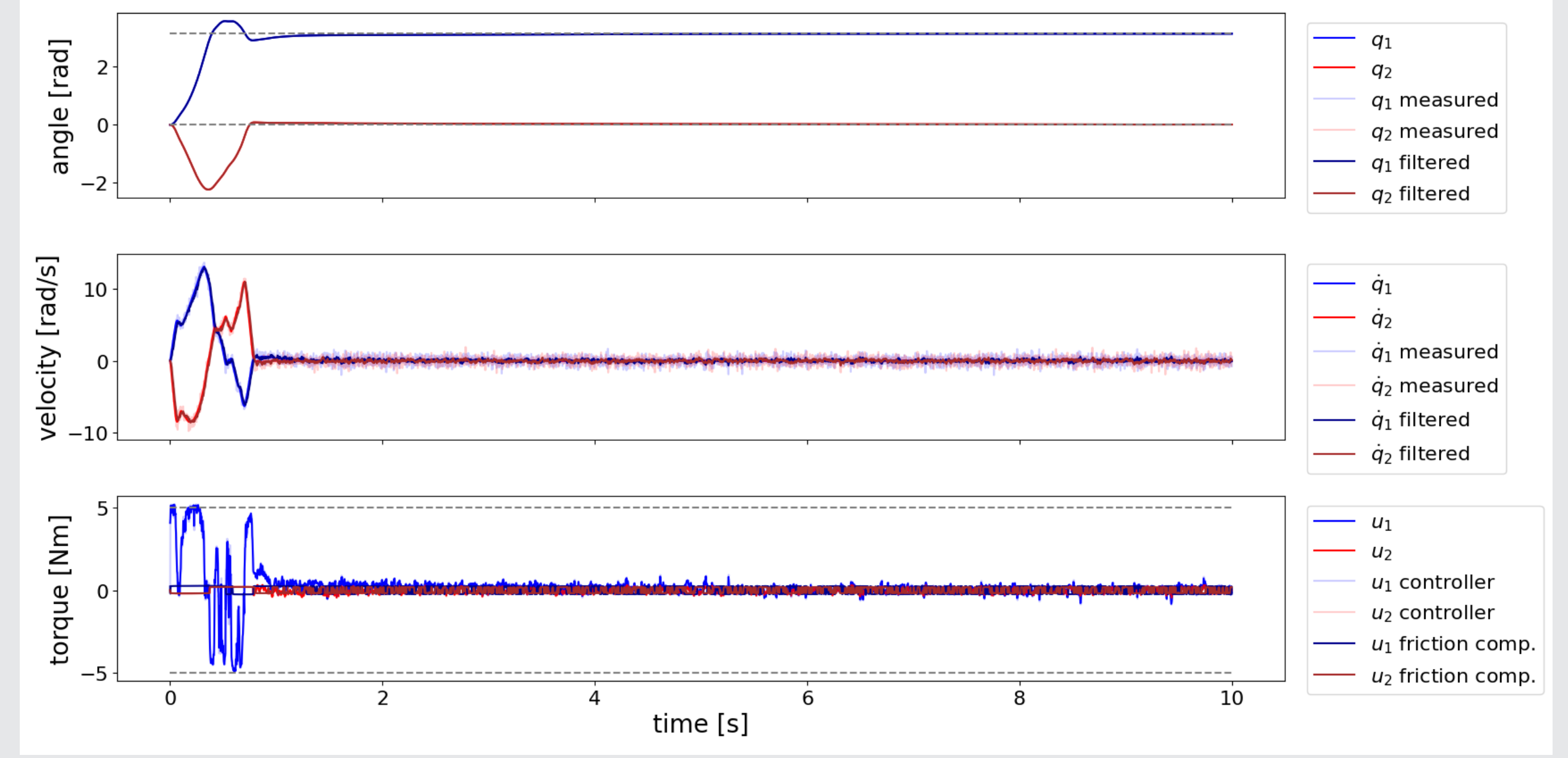
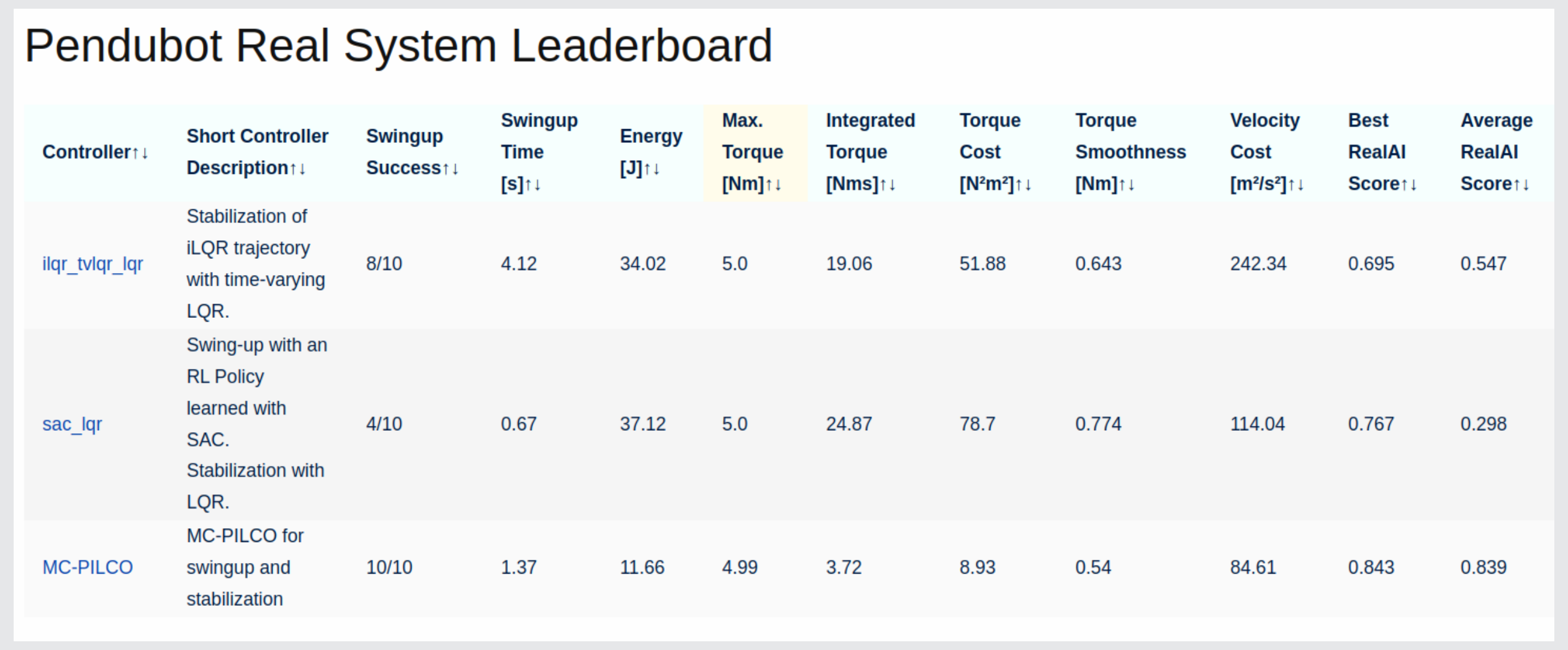


Figure: Successful simulation for pendubot

Our current approach to the problem involves creating a noisy environment that merges real-world features in the simulation. We then select a successfully trained agent from this noisy simulation and transfer it to the real system.

Real world experiment

Currently, SAC+LQR is only operational on the real pendubot system.



Pendubot Real System Leaderboard

Controller↑↓	Short Controller Description↑↓	Swingup Success↑↓	Swingup Time [s]↑↓	Energy [J]↑↓	Max. Torque [Nm]↑↓	Integrated Torque [Nms]↑↓	Torque Cost [N²m²]↑↓	Torque Smoothness [Nm]↑↓	Velocity Cost [m²/s²]↑↓	Best RealAI Score↑↓	Average RealAI Score↑↓
ilqr_tvlqr_lqr	Stabilization of iLQR trajectory with time-varying LQR.	8/10	4.12	34.02	5.0	19.06	51.88	0.643	242.34	0.695	0.547
sac_lqr	Swing-up with an RL Policy learned with SAC. Stabilization with LQR.	4/10	0.67	37.12	5.0	24.87	78.7	0.774	114.04	0.767	0.298
MC-PILCO	MC-PILCO for swingup and stabilization	10/10	1.37	11.66	4.99	3.72	8.93	0.54	84.61	0.843	0.839

Figure: Pendubot real system leaderboard

The success rate for the real system application is 40%, as recorded on the leaderboard. The positive aspect is the swift swing-up time of 0.67s and moderate energy consumption at 37.12J. However, the challenge lies in torque smoothness, measured at a low 0.298, likely a major factor in failed runs. Regrettably, the method is currently ineffective for the real acrobot setup.

Result and future work

In simulations, both pendubot and acrobot setups succeed. In real-world tests, only pendubot achieves a 40% success rate. The controller excels in swing-up time and moderate energy metrics, but torque-related challenges remain. We target effective sim-to-real transfer, particularly for the acrobot, and consider model-based RL like PILCO and extensions.

Acknowledgment

This work was supported by the federal state of Bremen for setting up the Underactuated Robotics Lab under Grant 201-342-04-2/2021-4-1.

References

[1] Jens Kober, J Andrew Bagnell, and Jan Peters. "Reinforcement learning in robotics: A survey". In: *The International Journal of Robotics Research* 32.11 (2013), pp. 1238–1274.
[2] Felix Wiebe et al. "An Open Source Dual Purpose Acrobot and Pendubot Platform for Benchmarking Control Algorithms for Underactuated Robotics". In: *IEEE Robotics and Automation Magazine* (2023). under review.
[3] Tuomas Haarnoja et al. "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor". In: *International conference on machine learning*. PMLR. 2018, pp. 1861–1870.
[4] N. Lehtomaki, N. Sandell, and M. Athans. "Robustness results in linear-quadratic Gaussian based multivariable control designs". In: *IEEE Transactions on Automatic Control* 26.1 (1981), pp. 75–93. DOI: 10.1109/TAC.1981.1102565.
[5] Lasse Maywald et al. "Co-optimization of Acrobot Design and Controller for Increased Certifiable Stability". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS-2022)*. Oct. 2022. DOI: 10.13140/RG.2.2.36436.07043.