The Journal of Open Source Software

Torque-limited simple pendulum: A toolkit for getting familiar with control algorithms in underactuated robotics

Felix Wiebe *[1], **Jonathan Babel**†[1], **Shivesh Kumar** [1], **Shubham Vyas**[1], **Daniel Harnack**[1], **Melya Boukheddimi**[2], **Mihaela Popescu**[2], **and Frank Kirchner**[1,2]

1 DFKI GmbH Robotics Innovation Center, Bremen, Germany **2** Working Group Robotics, University of Bremen, Bremen, Germany

DOI: 10.21105/joss.03884

Software

- Review
- Repository
- Archive

Editor: George K. Thiruvathukal

Reviewers:

- @ricopicone
- @jingnanshi

Submitted: 22 October 2021
Published: 01 June 2022

Summary

There are many, wildly different approaches to robotic control. Underactuated robots are systems for which it is not possible to dictate arbitrary accelerations to all joints. Hence, a controller cannot be used to override the system dynamics and force the system on a desired trajectory as it often is done in classical control techniques. A torque-limited pendulum is arguably the simplest underactuated robotic system and thus is a suitable system to study, test, and benchmark different controllers.

This repository describes the hardware (Computer-aided design (CAD) models, Bill Of Materials (BOM), etc.) required to build a physical pendulum system and provides the software (Unified Robot Description Format (URDF) models, simulation, and controller) to control it. It provides a setup for studying established and novel control methods, and targets students and researchers interested in studying underactuation.

Statement of need

This repository is designed to be used in education and research. It lowers the entry barrier for studying underactuation in real systems, which is often overlooked in conventional robotics courses. With this software package, students who want to learn about robotics, optimal control, or reinforcement learning can gain hands-on experience with hardware and software for robot control. The dualistic approach of describing software and hardware as well as the large spectrum of control methods are outstanding features of this package in comparison to similar software such as stable baselines (Raffin et al., 2021), open AI gym (Brockman et al., 2016), and Drake (Tedrake & the Drake Development Team, 2019). Results from real experiments are provided to ensure reproducibility and evaluate novel control methods.

Background

This project provides an easy accessible plant for the pendulum dynamics, which is built up from scratch and uses only standard libraries. The plant can be passed to a simulator object, which is capable of integrating the equations of motion and thus simulating the pendulum's motion forward in time. The simulator can perform Euler and Runge-Kutta integration and can also visualize the motion in a matplotlib animation. Furthermore, it is possible to interface a controller to the simulator that sends control a signal in form of a torque τ to the motor.

*co-first author
†co-first author

The pendulum has stable (downward configuration) and unstable (upward configuration) fixed points. A challenge from the control point of view is to swing the pendulum up to the unstable fixed point and stabilize the pendulum in that state, respecting the torque limitation.

Mechanical Setup

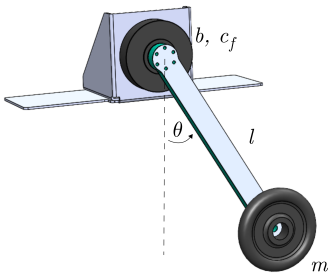

Figure 1: Simple Pendulum. The physical variables are introduced in Equation 1.

The pendulum (Figure 1) is constructed by mounting a motor to a fixed frame, attaching a rod to the motor, and a weight to the other end of the rod. The motor used in this setup is the AK80-6 actuator from T-Motor (CubeMars, 2021), which is a quasi direct drive with a gear ratio of 6:1 and a peak torque of 12 Nm at the output shaft.

Electrical Setup

The schematic below (Figure 2) displays the electrial setup of the testbench. A main PC is connected to a motor controller board (CubeMars_AK_V1.1, see CubeMars (2021)) mounted on the actuator. The communication takes place on a CAN bus with a maximum signal frequency of 1 Mbit/sec with the 'classical' CAN protocol. Furthermore, a USB to CAN interface is needed if the main PC doesn't feature a PCI CAN card. The actuator requires an input voltage of 24 Volts and consumes up to 24 Amps for peak torque. The power supply in our test setup is the EA-PS 9032-40 from Elektro-Automatik. The capacitor filters back EMF coming from the actuator and protects the power supply from high voltage peaks. An emergency stop button serves as additional safety measure.

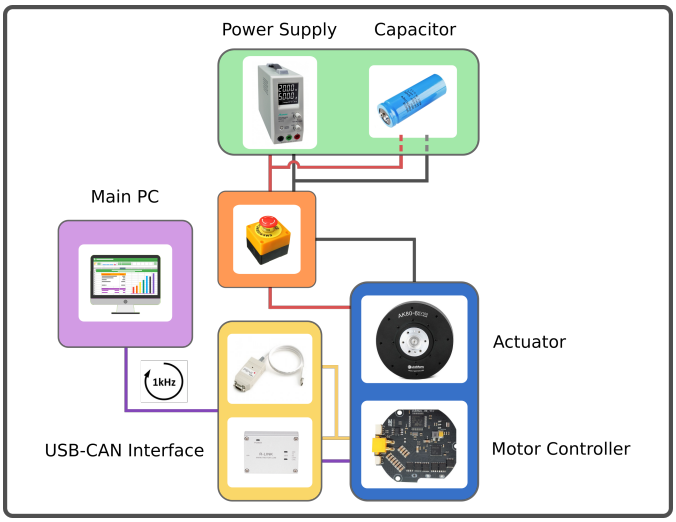

Figure 2: Electrical setup.

Pendulum Dynamics

The motions of a pendulum are described by the following equation of motion:

$$I\ddot{\theta} + b\dot{\theta} + c_f \text{sign}(\dot{\theta}) + mgl\sin(\theta) = \tau \tag{1}$$

where

- θ, $\dot{\theta}$, $\ddot{\theta}$ are the angular displacement, angular velocity and angular acceleration of the pendulum in counter-clockwise direction. $\theta = 0$ means the pendulum is at its stable fixed point (i.e., hanging down).
- I is the inertia of the pendulum. For a point mass: $I = ml^2$
- m mass of the pendulum
- l length of the pendulum
- b damping friction coefficient
- c_f coulomb friction coefficient
- g gravity (positive direction points down)
- τ torque applied by the motor

We provide a pendulum plant model that can be used for computing trajectories and policies without the actual hardware, simulating the execution of a controller, as well as simulating the system's response during real time control. Also, a system identification method is implemented that can reliably estimate the unknown pendulum parameters of the real setup.

Control Methods

The swing-up with a limited motor torque τ serves as a benchmark for various control algorithms. If the torque limit is set low enough, the pendulum is no longer able to simply go up to the unstable fixed point but instead, the pendulum has to swing and build up energy in the system.

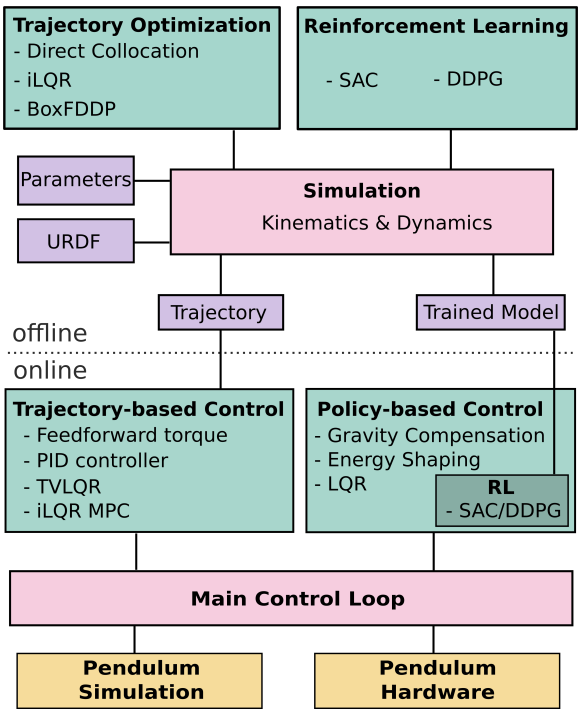

Figure 3: Control Software Structure.

The control methods that are currently implemented in this library (see also Figure 3) can be grouped in four categories:

Trajectory optimization tries to find a trajectory of control inputs and states that is feasible for the system while minimizing a cost function. The cost function can, for example, include terms that drive the system to a desired goal state and penalize the usage of high torques. The following trajectory optimization algorithms are implemented:

- Direct Collocation (Hargraves & Paris, 1987)
- Iterative Linear Quadratic Regulator (iLQR) (Weiwei & Todorov, 2004)
- Feasibility driven Differential Dynamic Programming (FDDP) (Mastalli et al., 2020)

The optimization is done with a simulation of the pendulum dynamics.

Reinforcement Learning (RL) can be used to learn a policy on the state space of the robot, which then can be used to control the robot. The simple pendulum can be formulated as a RL problem with two continuous inputs and one continuous output. Similar to the cost function in trajectory optimization, the policy is trained with a reward function. The following RL algorithms are implemented:

- Soft Actor Critic (SAC) (Haarnoja et al., 2018)
- Deep Deterministic Policy Gradient (DDPG) (Lillicrap et al., 2019)

Both methods are model-free, i.e., they use the dynamics of the system as a black box. Currently, learning is possible in the simulation environment.

Trajectory-based Controllers act on a precomputed trajectory and ensure that the system follows the trajectory properly. The trajectory-based controllers implemented in this project are:

- Feedforward torque
- Proportional-integral-derivative (PID) control
- Time-varying Linear Quadratic Regulator (TVLQR)
- Model Predictive Control (MPC) with iLQR

The Feedforward and PID controller are model-independent, while the TVLQR and iLQR MPC controllers utilize knowledge about the pendulum model. In contrast to the others, the iLQR MPC controller optimizes over a predefined horizon at every timestep.

Policy-based Controllers take the state of the system as input and output a control signal. In contrast to trajectory optimization, these controllers do not compute just a single trajectory. Instead, they react to the current state of the pendulum and because of this, they can cope with perturbations during the execution. The following policy-based controllers are implemented:

- Energy Shaping
- Linear Quadratic Regulator (LQR)
- Gravity Compensation

All of these controllers utilize model knowledge. Additionally, the control policies, obtained by one of the RL methods, fall in the category of policy-based control.

The implementations of direct collocation and TVLQR make use of Drake (Tedrake & the Drake Development Team, 2019), iLQR makes use of either the symbolic library of Drake or sympy, FDDP makes use of Crocoddyl (Mastalli et al., 2020), SAC uses stable-baselines3 (Raffin et al., 2021), and DDPG is implemented in TensorFlow (Abadi et al., 2016). The other methods use only standard libraries. This repository is designed to welcome contributions in form of novel optimization methods/controllers/learning algorithms to extend this list.

To get an understanding of the functionality of the implemented controllers, they can be visualized in the pendulum's state space. Example visualizations of the energy shaping controller and the policy learned with DDPG are shown in Figure 4.

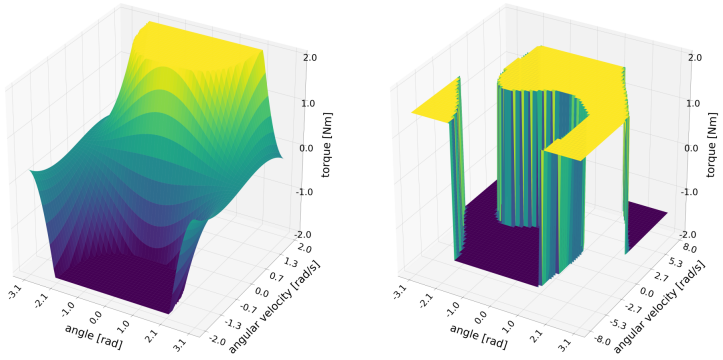

Figure 4: Energy Shaping Controller and DDPG Policy.

Furthermore, the swing-up controllers can be benchmarked in simulation, where how fast, efficient, consistent, stable and sensitive the controller is during the swing-up is evaluated. See Figure 5 for a comparison of the different controllers' benchmark results.

The benchmark criteria are:

- **Frequency**: The inverse of the time the controller needs to process state input and return a control signal (hardware dependent).
- **Swing-up time**: The time it takes for the controller to swing-up the pendulum from the lower fixed point to the upper fixed point.
- **Energy consumption**: The energy the controller uses during the swing-up motion and holding the pendulum in position afterwards.
- **Smoothness**: Measures how much the controller changes the control output during execution.
- **Consistency**: Measures if the controller is able to drive the pendulum to the unstable fixed point for varying starting positions and velocities.
- **Robustness**: Tests the controller abilities to recover from perturbations during the swing-up motions.
- **Insensitivity**: The pendulum parameters (mass, length, friction, inertia) are modified without using this knowledge in the controller.
- **Reduced torque limit**: The minimal torque limit with which the controller is still able to swing-up the pendulum.

The results shown in Figure 5 are the average of 100 repetitions for every controller and criterion. In the case of consistency, robustness, and insensitivity, the percentage refers to the ratio of successful swing-up motions of the 100 repetitions.

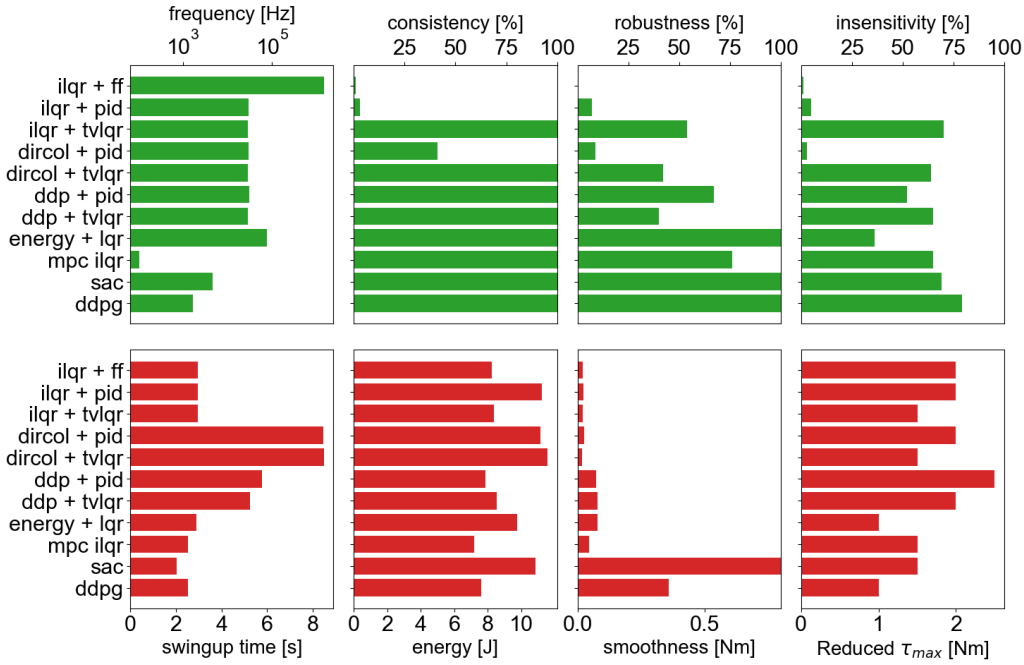

Figure 5: Benchmark results.

Trajectory optimization (iLQR, direct collocation, ddp) produces smooth trajectories, which swing up the pendulum relatively quickly. But they do require a trajectory-following control loop (PID, TVLQR) to make them more consistent, robust, and insensitive. This can become a problem for large deviations from the nominal trajectory. RL policies perform well on consistency, robustness, insensitivity, and are able to perform fast swing-up motions. Their drawback is that their output can fluctuate, which can result in rougher motions. The model predictive iLQR controller has an overall good performance but has the disadvantage that is it comparatively slow due to the optimization at every timestep. The energy shaping plus LQR controller, despite its simplicity, shows very satisfying results in all benchmark categories.

Acknowledgements

This work has been performed in the VeryHuman project funded by the German Aerospace Center (DLR) with federal funds (Grant Number: FKZ 01IW20004) from the Federal Ministry of Education and Research (BMBF) and is additionally supported with project funds from the federal state of Bremen for setting up the Underactuated Robotics Lab (Grant Number: 201-001-10-3/2021-3-2).

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., ... Zheng, X. (2016). *TensorFlow: Large-scale machine learning on heterogeneous distributed systems*. arXiv. https://doi.org/10.48550/ARXIV.1603.04467

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., & Zaremba, W. (2016). OpenAI Gym. *CoRR*, *abs/1606.01540*. https://doi.org/10.48550/ARXIV.1606.01540

CubeMars. (2021). *AK Series Actuator Driver Manual*. %7Bhttps://store.cubemars.com/images/file/20211201/1638329381542610.pdf%7D

Haarnoja, T., Zhou, A., Abbeel, P., & Levine, S. (2018). *Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor*. https://doi.org/10.48550/arXiv.1801.01290

Hargraves, C. R., & Paris, S. W. (1987). Direct Trajectory Optimization Using Nonlinear Programming and Collocation. *Journal of Guidance, Control, and Dynamics*, *10*(4), 338–342. https://doi.org/10.2514/6.1986-2000

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., & Wierstra, D. (2019). *Continuous Control with Deep Reinforcement Learning*. https://doi.org/10.48550/arXiv.1509.02971

Mastalli, C., Budhiraja, R., Merkt, W., Saurel, G., Hammoud, B., Naveau, M., Carpentier, J., Righetti, L., Vijayakumar, S., & Mansard, N. (2020). Crocoddyl: An Efficient and Versatile Framework for Multi-Contact Optimal Control. *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2536–2542. https://doi.org/10.1109/ICRA40945.2020.9196673

Raffin, A., Hill, A., Gleave, A., Kanervisto, A., Ernestus, M., & Dormann, N. (2021). Stable-Baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, *22*(268), 1–8. http://jmlr.org/papers/v22/20-1364.html

Tedrake, R., & the Drake Development Team. (2019). *Drake: Model-Based Design and Verification for Robotics*. https://drake.mit.edu

Weiwei, L., & Todorov, E. (2004). Iterative Linear Quadratic Regulator Design for Nonlinear Biological Movement Systems. *International Conference on Informatics in Control, Automation and Robotics*, 222–229. https://doi.org/10.5220/0001143902220229