DOI: DOIunavailable

Software

- Review
- Repository
- Archive

Editor: Pending Editor

Reviewers:

- @Pending Reviewers

Submitted: N/A
Published: N/A

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

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

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

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 on a simple torque-limited pendulum, and targets students and beginners of robotic control.

Statement of need

This repository is designed to be used in education and research. It targets lowering 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 make hands-on experiences with hardware and software for robot control. This dualistic approach of describing software and hardware is chosen to motivate experiments with real robotic hardware and facilitate the transfer between software and hardware. This dualism as well as the large spectrum of control methods are stand out features of this package in comparison to similar software such as open AI gym (Brockman et al., 2016) and Drake (Tedrake & Drake Development Team, 2019). To ensure reproducibility and evaluate novel control methods, not just the control methods' code but also results from experiments are provided.

[*]co-first author
[†]co-first author

Background

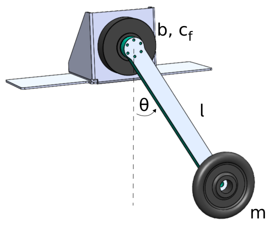

Figure 1: Simple Pendulum.

A pendulum (Figure 1) is constructed by mounting a motor to a fixed frame, attaching a rod to the motor and attaching a weight to the other end of the rod. The motor used in this setup is the AK80-6 actuator from T-Motor, 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) mounted on the actuator (AK80-6 from T-Motor). The communication takes place on a CAN bus with a maximum signal frequency of 1Mbit/sec with the 'classical' CAN protocol. Furthermore, a USB to CAN interface is needed, if the main PC doesn't have a PCI CAN card. The actuator requires an input voltage of 24 Volts and consumes up to 24 Amps under full load. A power supply that is able to deliver both and which is used in our test setup is the EA-PS 9032-40 from Elektro-Automatik. A capacitor filters the backEMF coming from the actuator and therefore protects the power supply from high voltage peaks. An emergency stop button serves as additional safety measure.

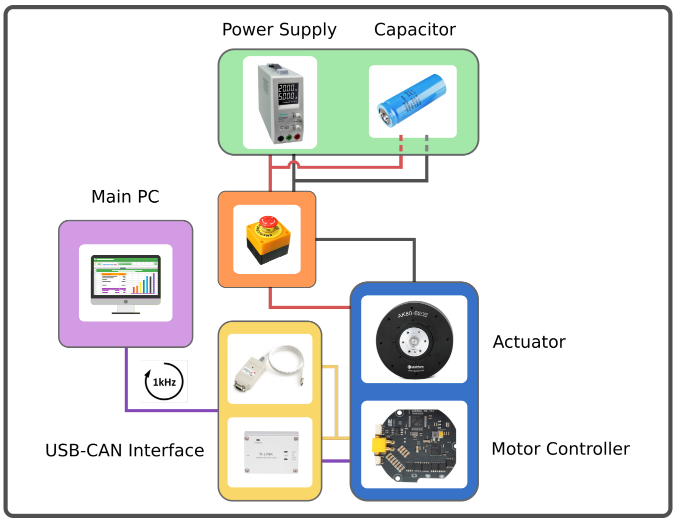

Figure 2: Electrical setup.

Pendulum Dynamics

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

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

where

- θ, $\dot{\theta}$, $\ddot{\theta}$ are the angular displacement, angular velocity and angular acceleration of the pendulum. $\theta = 0$ means the pendulum is at its stable fixpoint (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

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 which sends control a signal in form of a torque τ to the motor.

The pendulum has two fixpoints, one of them being stable (the pendulum hanging down) and the other being unstable (the pendulum pointing upwards). A challenge from the control point of view is to swing the pendulum up to the unstable fixpoint and stabilize the pendulum in that state.

Parameter Identification

The rigid-body model dervied from a-priori known geometry as described (Siciliano et al., 2009) by has the form

$$\tau(t) = \mathbf{Y}\left(\theta(t), \dot{\theta}(t), \ddot{q}(t)\right)\ \lambda,$$

where $\lambda \in \mathbb{R}^{12n}$ denotes the parameter vector with n sets of parameters λ_i,

$$\lambda_i = \left(m_i\ m_i c_{x,i}\ m_i c_{y,i}\ m_i c_{z,i}\ I_{xx,i}\ I_{xy,i}\ I_{xz,i}\ I_{yy,i}\ I_{yz,i}\ I_{zz,i}\ F_{c,i}\ F_{v,i}\right)^T$$

Two additional parameters for Coulomb and viscous friction are added to the model, $F_{c,i}$ and $F_{v,i}$, in order to take joint friction into account (Bargsten et al., 2016). For a reference trajectory sampled an *identification matrix* Φ can be created. The required torques for model-based control can be measured using stiff position control and closely tracking the reference trajectory. A sufficiently rich, periodic, band-limited excitation trajectory are obtained by modifying the parameters of a Fourier-Series as described by (Swevers et al., 2007). The dynamic parameters $\hat{\lambda}$ are estimated through least squares optimization between measured torque and computed torque :

$$\hat{\lambda} = \underset{\lambda}{\text{argmin}}\left((\Phi\lambda - \tau_m)^T(\Phi\lambda - \tau_m)\right).$$

Control Methods

The swing-up challenge 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 fixpoint but instead the pendulum has to swing and built up energy in the system.

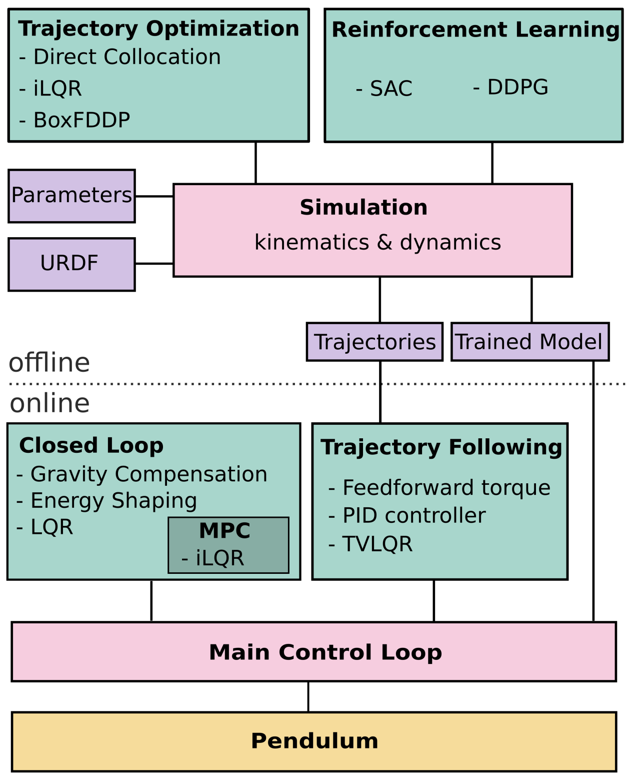

Figure 3: Control Software Structure.

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 which 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)

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

- Feedforward torque
- Proportional-integral-derivative (PID) control
- Time-varying Linear Quadratic Regulator (TVLQR)

Closed Loop Controllers or feedback controllers take the state of the system as input and ouput a control signal. Because they are able to react to the current state, they can cope with perturbations during the execution. The following feedback controllers are implemented:

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

- Model Predictive Control (MPC) with iLQR

Reinforcement Learning (RL) can be used to learn a policy on the state space of the robot. The policy, which has to be trained beforehand, receives a state and outputs a control signal like a feedback controller. The simple pendulum is 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)

The implementations of direct collocation and TVLQR make use of drake (Tedrake & Drake Development Team, 2019), iLQR only makes use of the symbolic library of drake, FDDP is makes use of Crocoddyl (Mastalli et al., 2020), SAC uses the stable-baselines3 (Raffin et al., 2019) implementation and DDPG is implemented in tensorflow (Abadi et al., 2015). 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 Figure 4.

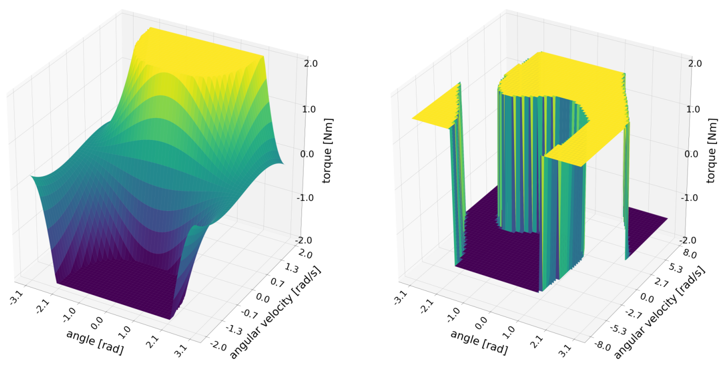

Figure 4: Energy Shaping Controller and DDPG Policy.

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

The benchmark criteria are:

- **Speed** : The inverse of the time the controller needs to process state input and return a control signal (hardware dependent).
- **Swingup time** : The time it takes for the controller to swing-up the pendulum from the lower fixpoint to the upper fixpoint.
- **Energy consumption**: The energy the controller uses during the swingup 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 fixpoint for varying starting positions and velocities.
- **Robustness**: Tests the controller abilities to recover from perturbations during the swingup 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.

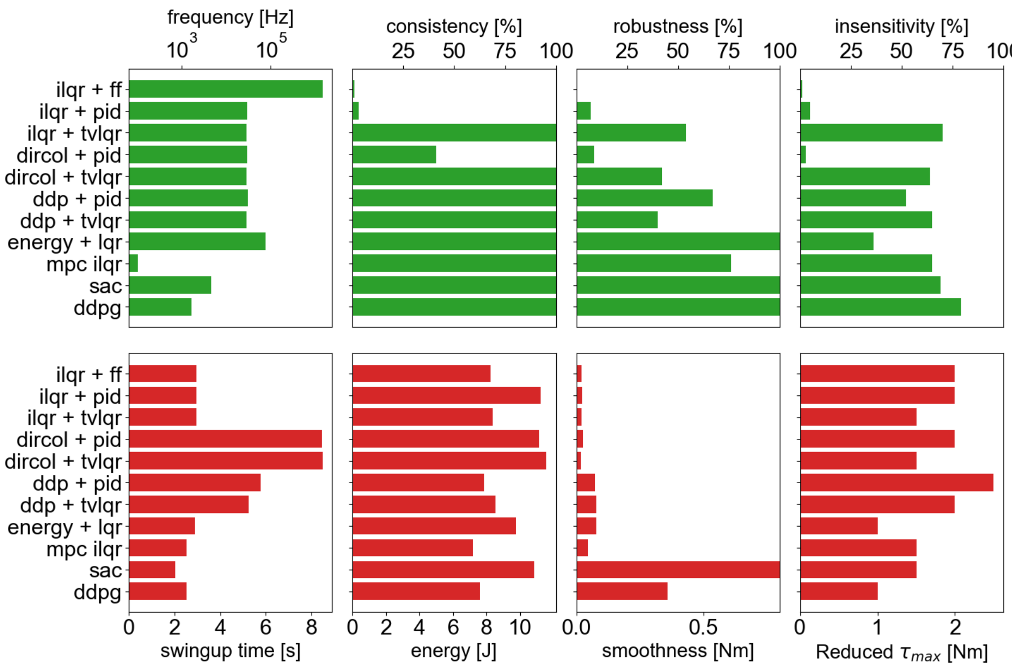

Figure 5: Benchmark results.

Trajectory optimization (iLQR, direct collocation, ddp) produces smooth trajectories, which swingup the pendulum relatively quick. 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 perfrom fast swingup 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. (2015). *TensorFlow: Large-scale machine learning on heterogeneous systems.* https://www.tensorflow.org/

Bargsten, V., Gea Fernández, J. de, & Kassahun, Y. (2016). *Experimental robot inverse dynamics identification using classical and machine learning techniques* (I. S. on Robotics, Ed.).

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., & Zaremba, W. (2016). OpenAI gym. *CoRR*, *abs/1606.01540*. http://arxiv.org/abs/1606.01540

Haarnoja, T., Zhou, A., Abbeel, P., & Levine, S. (2018). *Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor*. http://arxiv.org/abs/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.

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*. http://arxiv.org/abs/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., Ernestus, M., Gleave, A., Kanervisto, A., & Dormann, N. (2019). Stable Baselines3. In *GitHub repository*. https://github.com/DLR-RM/stable-baselines3; GitHub.

Siciliano, B., Sciavicco, L., Villani, L., & Oriolo, G. (2009). *Robotics*. Springer London. https://doi.org/10.1007/978-1-84628-642-1

Swevers, J., Verdonck, W., & De Schutter, J. (2007). Dynamic model identification for industrial robots. *IEEE Control Systems*, *27*(5), 58–71. https://doi.org/10.1109/MCS.2007.904659

Tedrake, R., & Drake Development Team, the. (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.