MITSUBISHI ELECTRIC RESEARCH LABORATORIES
https://www.merl.com

# Model-Based Learning Controller Design for a Furuta Pendulum

Nikovski, Daniel; Yerazunis, William S.; Goldsmith, Abraham

TR2023-080 July 04, 2023

**Abstract**

We present a method for designing and tuning controllers for the problem of swing-up and stabilization of a Furuta pendulum. The method is based on suitable param- eterization of a family of controllers and the application of Bayesian optimization to their tuning with minimal interaction with the physical system. Unlike traditional controller design methodologies, the method does not require the derivation of an exact physical model of the controlled plant, thus saving significant design time and effort. Furthermore, the method has much more favorable sample complexity than most policy optimization methods proposed in the field of reinforcement learning.

*9th International Conference on Control, Decision and Information Technologies (CoDIT) 2023*



Mitsubishi Electric Research Laboratories, Inc.
201 Broadway, Cambridge, Massachusetts 02139

# Model-Based Learning Controller Design for a Furuta Pendulum

Daniel Nikovski, William Yerazunis and Abraham Goldsmith†

***Abstract*— We present a method for designing and tuning controllers for the problem of swing-up and stabilization of a Furuta pendulum. The method is based on suitable parameterization of a family of controllers and the application of Bayesian optimization to their tuning with minimal interaction with the physical system. Unlike traditional controller design methodologies, the method does not require the derivation of an exact physical model of the controlled plant, thus saving significant design time and effort. Furthermore, the method has much more favorable sample complexity than most policy optimization methods proposed in the field of reinforcement learning.**

## I. Introduction

The field of automatic control relies on a wide variety of controller design methodologies that address various aspects of the controlled plant, such as non-linearity, under-actuation, unstable open-loop dynamics, etc. For a large number of simpler control applications, single-input single-output (SISO) proportional-integral-derivative (PID) controllers are sufficient for satisfactory performance and have been used widely in practice. Although sometimes the gains of a PID controller can be tuned without detailed knowledge of the dynamics of the plant, having such knowledge is usually advantageous in designing the controller. Other, more advanced controller design methodologies such as Linear Quadratic Regulator (LQR) design do require full knowledge of the system dynamics in the form of a linear state space model. For many systems, deriving such models in the needed state-space form requires extensive physical and mathematical modeling that is difficult, laborious, error-prone, and often resulting in only imprecise approximations to the real system dynamics.

A substantially different approach to system modeling and controller design consists of using machine learning methods to identify suitable system models and/or devise suitable control laws. The big advantage of this approach are the potential savings in human time and effort for system modeling and controller design. Recent major successes in deep reinforcement learning (DRL) in the field of computer games and other hard decision problems have opened the possibility for autonomous controller design based on autonomous self exploration and experimentation with the controlled system. However, unlike computer games that allow a very large number of experimental trials, physical system are relatively slow to operate and only a limited number of experimental trials are possible. Furthermore, completely random exploration could be dangerous and harmful for the physical system or its operators. Consequently, currently there is a significant need to develop learning controller design methodologies that can operate on real systems within reasonable time and in a safe manner [1].

One possibility, explored in this paper, is to use as much as possible knowledge about the general structure of a suitable controller, introduce a suitable parametric form for this controller, and use suitable learning and optimization tools to find good values for the parameters. This leverages knowledge that is often readily available in the research literature, without incurring the large cost of detailed physical design for the concrete system to be controlled. Moreover, the use of learning and optimization technologies can potentially result in controllers whose performance is superior to that of ones based on manually derived physical models. We have been investigating this approach using a known difficult benchmark problem in the field of control: the swing-up and stabilization of a rotary pendulum.

## II. Furuta Pendulum Control

The rotary pendulum, also known as the Furuta pendulum (FP) after the name of its inventor, has been widely used to investigate advanced controller design [2]. It consists of two links, an arm and a pendulum (Fig. 1), such that the arm is actuated by means of applied torque $\tau$ around the vertical axis $Z$, and the pendulum rotates freely in a plane perpendicular to the arm, without any actuation. The state of the FP $x = [\theta_1, \dot{\theta}_1, \theta_2, \dot{\theta}_2]^T$ is described by the arm and pendulum angles $\theta_1$ and $\theta_2$ and their angular velocities $\dot{\theta}_1$ and $\dot{\theta}_2$.

Swinging up the pendulum from its low stable equilibrium position $\theta_2 = 0$ to the upper unstable equilibrium position $\theta_2 = \pi$, stabilizing it there, and possibly also bringing the arm of the pendulum to a desired angle $\theta_1 = \theta_d$ while balancing the pendulum up, define a class of difficult control problems. Basic PID control applied to each state variable independently is not a suitable choice for swing-up and stabilizing controllers, especially in under-actuated systems. PID controllers, by their nature, always try to minimize the control error. However, when the system has a limit $\tau_{max}$ on the torque applied to the arm, $|\tau| \leq \tau_{max}$, this strategy will not succeed; instead, multiple swings of the pendulum are necessary, during some of which the pendulum must be moving away from its desired upright position. This behavior cannot be implemented by means of a PID controller that will actively oppose moving away from the target. Similarly, when the pendulum is already in its upright position, and the arm needs to move to a different desired angle while balancing the pendulum, the right motion is to first move the arm away from that angle, in order to tilt the pendulum

†All authors are with Mitsubishi Electric Research Laboratories (MERL), Cambridge, MA, USA 02139 {nikovski,yerazunis,goldsmith}@merl.com

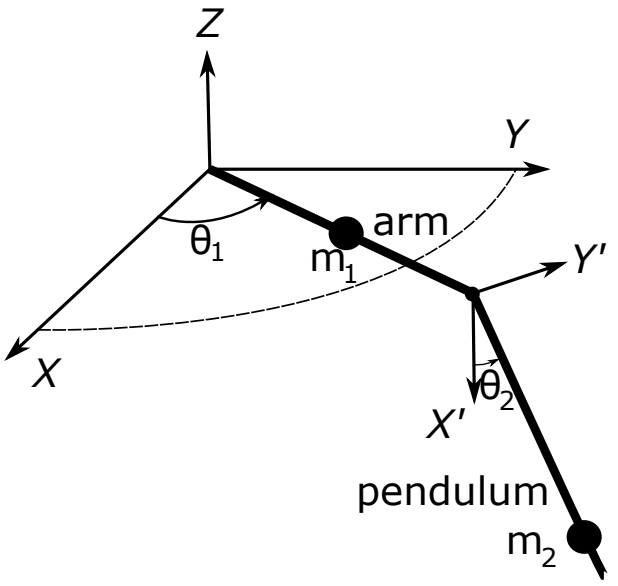


Fig. 1. Furuta pendulum.

in the right way, thus briefly increasing the control error. This is a behavior that a decoupled PID controller consisting of two (or four) SISO controllers, for the arm and pendulum respectively (plus possibly their angular velocities), would not exhibit.

Certainly, other well established control design methods such as linear quadratic regulator (LQR) design can be employed to solve the stabilization problem. In fact, the original paper that introduced the FP did describe an LQR stabilizing controller based on a physical model derived by means of Lagrangian mechanics and carefully calibrated to match the real experimental system [2]. As pointed out above, the disadvantage of this methodology is in the laborious derivation and calibration of the physical model. In later work, a class of energy manipulating controllers was proposed for swinging up torque-limited pendulums [3]. Their operation, too, depends critically on knowing precisely the inertial properties of the system in order to avoid failing to reach or overshooting the unstable upright equilibrium of the pendulum.

Learning-based controller design has been applied to the FP, too. Although applying model-free DRL to learning an entire controller for both the swing-up and stabilization parts of the problem is possible in principle, at least in simulation and similarly to how DRL has been widely used for cart-pole balancing, the required number of trials is much too high for practical use on a real system. Using recurrent neural networks with RL for the design of the stabilizing controller only has been described in [4], in conjunction with a more traditional energy-manipulation swing-up controller. Model-based DRL is a much more viable approach, and has recently been applied successfully to FP control in conjunction with a method for learning accurate models of the system dynamics by means of Gaussian processes [5]. The success of this approach, though, depends critically on the accuracy of the learned dynamics model, as all controller tuning is done by means of the learned model.

## III. Learning of Parameterized Controllers

A logical and attractive middle ground between full physics-based controller design and full learning-based one is the approach based on identifying the general structure of the control law from physical principles and past controller design practice, parameterizing it suitably to define a family of possible controllers, and tuning these parameters by means of self-experimentation with the real system, much like other learning controllers do. This kind of controller remains essentially a learning controller, similar to controllers identified by means of DRL. The difference is that this controller has a very strong inductive bias that allows it to learn quickly. Arguably, humans have various strong inductive biases in terms of the properties of the world surrounding them, such as continuity and persistence of objects' form and mass, regularity of motion, continuity of visual fields, etc., that help them learn new concepts and skills very quickly, from only few examples or experimental trials. In terms of controller design, similar strong inductive bias can be injected into the learning process by defining a suitable parametric form of the controller.

### A. Controller Parameterization

For the FP and similar applications, the wealth of the research literature suggests suitable parametric forms. Successful applications of LQR design to FP stabilization (such as [2]) showed that a full-state feedback controller of the form $\tau = -Kx$ can be constructed by choosing suitable feedback gains $K = [k_1, k_2, k_3, k_4]$. Similarly, an energy manipulation swing-up controller of the form used in [3] and [4] can take the form

$$\tau = \begin{cases} -\tau_0 \dot{\theta_2} \cos\theta_2, & \text{if } |\theta_2| \leq \epsilon \\ 0, & \text{otherwise,} \end{cases}$$

for a suitably chosen energy-increasing torque $0 < \tau_0 \leq \tau_{max}$ acting only in a neighborhood of size $\epsilon_1$ around the lower stable equilibrium $\theta_2 = 0$.

As the control scheme involves two controllers, one for swing-up and another for stabilization, it is necessary to also include a decision rule that specifies which controller will be active at a given time. Because the feedback gains $K$ match the linearization of the system's nonlinear dynamics around the upper unstable equilibrium $\theta_2 = \pi$, a stabilization controller with these gains is effective only in a neighborhood of size $\epsilon_2$ of the pendulum angle around this unstable equilibrium. Thus, the stabilization controller can be activated when the pendulum angle enters the interval $\theta_2 \in [\pi - \epsilon_2, \pi + \epsilon_2]$. Similar to the other parameter values, $\epsilon_2$ can be subject to optimization, too.

### B. Parameter Tuning by Means of Bayesian Optimization

Once the parameter vector $p = [k_1, k_2, k_3, k_4, \tau_0, \epsilon_1, \epsilon_2]$ has been defined, the question becomes how to find optimal values for the parameters in a minimal time and with

minimal interaction with the physical system. A suitable class of optimization methods with increasing popularity are Bayesian optimization (BO) algorithms [6]. BO can optimize a performance measure without having an analytical objective function that describes how this performance measure depends on the decision variables (the parameter vector $p$, in this case). This setting matches well the controller design for the FP: although it is relatively easy to specify a performance measure to evaluate the success of a particular controller, it is not easy to describe the dependence of this performance measure on the controller parameters. A possible performance measure could be defined easily in terms of the cumulative controller error over a period $T$ that is longer than the time sufficient for successful swing-up of the pendulum:

$$C(p) = \sum_{k=1}^{n} (\theta_2[k] - \pi)^2 + (\theta_1[k] - \theta_d)^2, \qquad (1)$$

where $n = T/\Delta t$ is the number of control steps until time $T$ and $\Delta t$ is the control time step. The second term can be omitted, if the angle of the arm is not important, for an easier version of the swing-up and stabilization problem.

In addition to the objective function defined in Eq. 1, Bayesian optimization algorithms require suitable bounds for the decision variables in order to be effective in finding their optimal values. Again, the research literature suggests that the gains for the arm position and velocity should be negative, corresponding to *positive* feedback on the arm error, thus pushing it away from the target angle $\theta_d$. Conversely, the gains for the pendulum's position and velocity should be positive, resulting in negative feedback on the pendulum error, thus correcting deviations of the pendulum from the upright equilibrium. (The relative magnitude of the gains of the two loops, as well as the state variables, will determine which of the feedback loops will prevail in moving the arm in one direction or another.) This understanding suggests what the boundaries of the box constraints on the gains should be. Similarly, the neighborhood limits $\epsilon_1$ and $\epsilon_2$ can be specified to be within the interval $[0, \pi]$, but in practice, the neighborhood in which the stabilization controller is effective has usually been reported to be within $\pi/10$ radians from the unstable equilibrium (e.g. [4]), so this value can be used as the upper limit for $\epsilon_2$.

## IV. Experimental Setup

In order to verify the effectiveness of the proposed method, we have built a physical set-up of an FP, shown in Fig. 2, as well as a simulation model corresponding to it shown in Fig. 3. The physical model is equipped with a servo motor MELFA HG-KR23K driven by a servo amplifier MELFA MR-J4-20A1-RJ in torque control mode. The pendulum is of length $600mm$ and weight $150g$, and is suspended from a pivot point on the arm located $290mm$ from the arm's axis of rotation.

The simulation model has been implemented in the physics engine MuJoCo that has been popular in research on learning controllers [7]. Note that the simulation model defines only the rigid bodies comprising the two links of the pendulum, the bearings implementing the pendulum joint, and the encoder of the pendulum that is mounted on the arm, in terms of their geometric properties, relative positions to each other, and their masses. The model is defined in an XML file. This kind of modeling is much faster and easier to do than deriving full equations of motion from first principles.

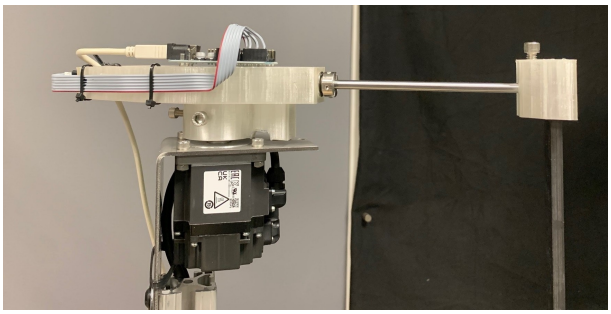

Fig. 2. A physical FP implemented with a servo motor driven in direct torque-control mode.

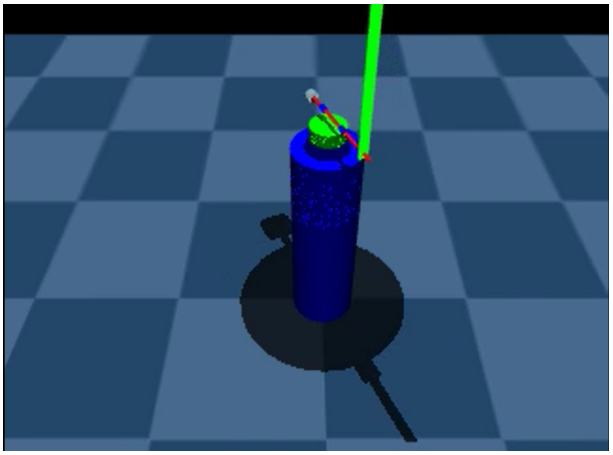

Fig. 3. A model created in the physics engine MuJoCo. It consists of bodies whose geometric and inertial parameters, as well as their relative geometric positions, match those of the main components of the real FP: the pendulum, the axle of the arm, the ball bearings implementing the joint of the pendulum, the encoder of the pendulum axis and its coupler to the axis.

## V. Experimental Verification

So far, we have been successful in applying the proposed method in simulation, on the MuJoCo model. Using the *bayesian-optimization* package in Python [8], we were able to successfully discover values for the parameter vector that can bring up the FP from its resting stable position, with initial position of the arm at $\theta_1 = 0$, and identical desired position in the upper unstable equilibrium, $\theta_d = 0$. That is, the goal was to swing up the pendulum so that it is balanced exactly above where it started from.

The Bayesian optimization algorithm was able to discover good values for the stabilizer gains $K$ and swing-up torque $\tau_0$ if we set the two neighborhoods to reasonable values, namely $\epsilon_1 = 0.2rad$ and $\epsilon_2 = 0.1rad$. For these neighborhoods, the

BO routine with $1,000$ random initial points followed by $10$ improvement iterations was able to discover the following parameters resulting in successful swing-up and stabilization of both the arm and the pendulum at their desired angles. The discovered pushing torque was $\tau_0 = 10.97Nm$ and the feedback gains were $K = [-2.548, -1.485, 25.79, 3.358]$. Fig. 4 shows that the discovered solution used two swings of the pendulum. When there are no limits on the torque, the time-optimal solution that would minimize the cost $C(p)$ consists of a single swing, which would be the global minimum of the optimization problem. In our case, the algorithm discovered the second-best local minimum that used two swings.

## VI. Conclusion and Future Work

We presented a method for designing controllers based on parameterizing a controller with suitable structure by a vector of parameters and using Bayesian optimization to find good values for these parameters. In the future, we plan to apply the same approach to the real physical system. Of significant interest is whether good solutions can be discovered in simulation and optimization on the real system hot-started with them on the real system. This could be done by defining relatively tight box constraints for the parameters for Bayesian optimization on the real system, centered around the solution found on the simulated system. Another approach is to optimize controller parameters by introducing multiple deliberate variations in the model used in simulation in order to obtain a robust policy, following the practice of domain randomization.

A complementary technique would be to use initial runs on the real system to calibrate the inertial and friction parameters of the simulated one. Bayesian optimization can be helpful for this task, too, with a cost function that reflects the discrepancy between the behavior of the real and simulated systems for identical inputs.

Other methods for finding suitable values for the parameters can be explored, too. Of particular relevance are policy gradient methods that have been shown to operate successfully on similar parameterized controllers for other robotic applications [1]. A combination of using a DRL method with high sample complexity in simulation and a low-complexity optimization method on the real system might be possible, too, if the policy discovered by the DRL method can be distilled into a controller with few parameters whose values can be tuned on the real system.

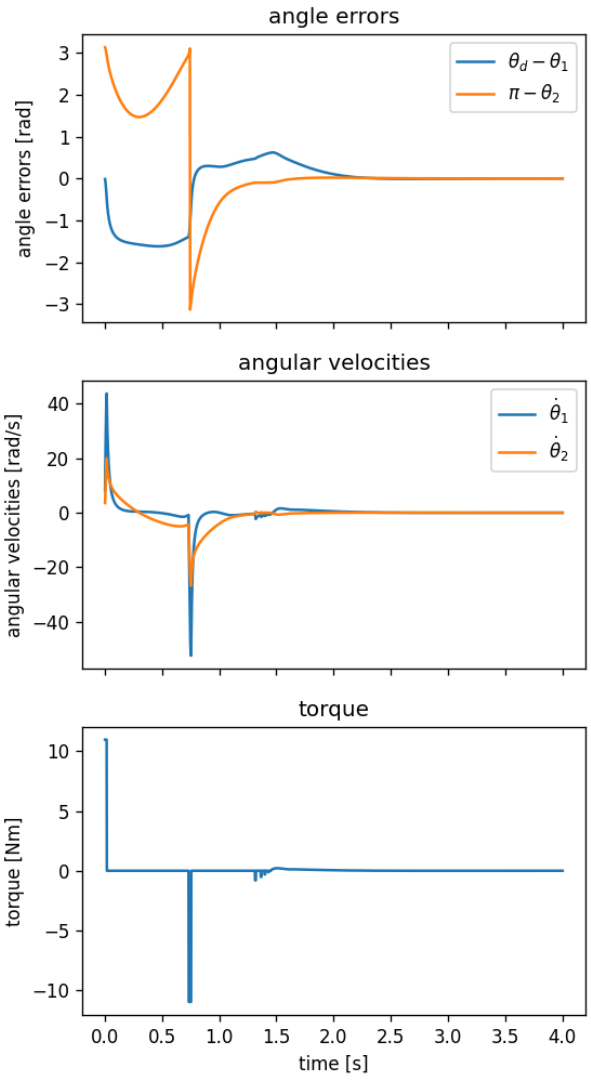


Fig. 4. Controller performance on FP swing-up and stabilization in simulation, using gains and swing-up torque discovered by means of BO. Top: control error in the angles of the arm and pendulum, in terms of which the cost is defined. Convergence to zero error and thus zero incremental cost is evident. The arm's angle plot shows that two swings are necessary. Middle: velocities converging to zero. Bottom: the controller gives two pushes in opposite direction at the bottom of the swing, pumping energy, after which the pendulum coasts on momentum to the neighborhood of the upper unstable equilibrium. The stabilizing controller then brings the arm to its desired angle $t_d$ while balancing the pendulum in the upright position.

## References


[1] J. Peters and S. Schaal, "Policy gradient methods for robotics," in *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2006, pp. 2219–2225.

[2] K. Furuta, M. Yamakita, S. Kobayashi, and M. Nishimura, "A new inverted pendulum apparatus for education," in *Advances in Control Education 1991*. Elsevier, 1992, pp. 133–138.

[3] K. J. Åström and K. Furuta, "Swinging up a pendulum by energy control," *Automatica*, vol. 36, no. 2, pp. 287–295, 2000.

[4] B. Allotta, L. Pugi, and F. Bartolini, "Reinforcement neural network for the stabilization of a Furuta pendulum," in *Proceedings of EUCOMES 08: The Second European Conference on Mechanism Science*. Springer, 2009, pp. 287–294.

[5] F. Amadio, A. Dalla Libera, R. Antonello, D. Nikovski, R. Carli, and D. Romeres, "Model-Based Policy Search Using Monte Carlo Gradient Estimation With Real Systems Application," *IEEE Transactions on Robotics*, vol. 38, no. 6, pp. 3879–3898, 2022.

[6] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. De Freitas, "Taking the human out of the loop: A review of Bayesian optimization," *Proceedings of the IEEE*, vol. 104, no. 1, pp. 148–175, 2016.

[7] E. Todorov, T. Erez, and Y. Tassa, "MuJoCo: A physics engine for model-based control," in *International Conference on Intelligent Robots and Systems*. IEEE, 2012, pp. 5026–5033.

[8] F. Nogueira, "Bayesian Optimization: Open source constrained global optimization tool for Python," 2014–. [Online]. Available: https://github.com/fmfn/BayesianOptimization