Compliant Robot Behavior Using Servo Actuator Models Identified by Iterative Learning Control

Max Schwarz and Sven Behnke

Rheinische Friedrich-Wilhelms-Universität Bonn
Computer Science Institute VI: Autonomous Intelligent Systems
Friedrich-Ebert-Allee 144, 53113 Bonn, Germany
max.schwarz@uni-bonn.de, behnke@cs.uni-bonn.de

Abstract. System parameter identification is a necessary prerequisite for model-based control. In this paper, we propose an approach to estimate model parameters of robot servo actuators that does not require special testing equipment. We use Iterative Learning Control to determine the motor commands needed to follow a reference trajectory. To identify parameters, we fit a model for DC motors and friction in geared transmissions to this data using a least-squares method. We adapt the learning method for existing position-controlled servos with proportional controllers via a simple substitution. To achieve compliant position control, we apply the learned actuator models to our humanoid soccer robot NimbRo-OP. The experimental evaluation shows benefits of the proposed approach in terms of accuracy, energy efficiency, and even gait stability.

1 Introduction

DC servo motors are popular actuators in the field of robotics because of their ease of use and low cost. Traditional control methods often ignore the dynamics of the motor, in particular friction forces, and compensate the loss of knowledge about the system through sensory feedback. While it is possible to reach very small position errors with this method, high-gain position control often results in undesirable behavior like stiffness and oscillations (limit cycles).

The research field of humanoid soccer robots places unique demands on the performance of robot joints. Having the ability to perform highly dynamic motions is more important than accurate setpoint tracking at low speeds. These motions require considerable torques and moments and can be dangerous to the joint—in particular to the gear—if not executed properly.

The use of motor and friction models enables the controller to demand exactly the torque needed to follow a position trajectory. In addition to minimizing energy consumption, this leads to a more compliant motion of the robot in the face of unexpected obstacles or perturbations—a feature that is important in many robot applications.

Determining motor model coefficients can be a daunting task, since specific test runs in controlled situations are needed. Often the motor cannot remain in the robot for parameter measurement. Additionally, special test setups might

S. Behnke et al. (Eds.): RoboCup 2013, LNAI 8371, pp. 207–218, 2014.

be needed to produce fixed load conditions. These problems call for a simpler identification method, which is the main objective of this work.

In this paper, we define models for DC motors and friction in gear transmissions. We apply Iterative Learning Control [1] to follow a reference trajectory. From the resulting motor commands, we identify the model parameters. We evaluate this learning and identification process on our NimbRo-OP humanoid soccer robot [2]. Finally, we apply the model for feed-forward control during full-body walking motions.

2 Related Work

Friction effects in robot joints have been thoroughly investigated and characterized. Waiboer et al. [3] successfully modeled friction forces in robotic joints as friction between lubricated discs in a rolling-sliding contact. For parameter estimation, a four step least-squares fitting involving the hand-tuning of Stribeck parameters is needed.

A smaller version *(AX-12+)* of the actuator used for evaluation *(MX-106)* has been modeled before [4] with a similar friction model. Special test setups were used to determine viscous and static friction constants. The Stribeck parameters were also hand-tuned.

A few approaches for online learning of friction compensation torques exist. Kim et al. [5] apply reinforcement learning with a neural network to control a 1-DOF system with changing friction parameters.

Iterative Learning Control has been proposed as a method for friction compensation before. Liu [6] used a PD type iterative control for learning torque commands to overcome friction effects on a fixed trajectory. However, generalization to other trajectories or operating conditions was not attempted.

In contrast to the existing work, the proposed system identification method does not require special test benches or isolation of the actuator. The parameters are estimated using a trajectory that is relevant to the robot's general operation. This ensures good results in important position, velocity, and acceleration ranges. While the Stribeck curve parameters still might need to be tuned by hand, all other parameters are calculated using a single linear optimization.

Of particular interest here is the direct modeling of DC motors, which has not been at the center of scientific attention since most robot actuators are controlled by position or torque, hiding the internal workings of the servo motor behind a controller interface.

3 Modeling Robot Joints

3.1 Motor Model

The motor model we are considering here describes a simplified ideal DC motor. This assumption is a good one for most robotics actuators, since they contain high-quality DC motors.

A simple model for an ideal DC motor can be derived by considering the power balance present in the motor at a constant voltage U:

$$P_{el} = P_{mech} + P_J, \tag{1}$$

where P_{el} is the electrical power consumed, P_{mech} denotes the mechanical output power and P_J corresponds to the Joule heating losses in the motor winding. Further substitution yields

$$UI = \omega\tau + RI^2 \tag{2}$$

$$\Leftrightarrow\ U = \omega\frac{\tau}{I} + RI, \tag{3}$$

where U is the voltage applied to the motor, I is the current flowing through the winding, ω and τ are the present angular velocity and torque, respectively, and R denotes the motor winding resistance.

The torque constant k_τ describes the relationship of τ and I:

$$\tau = k_\tau I, \tag{4}$$

which gives

$$U = \omega k_\tau + \frac{R}{k_\tau}\tau. \tag{5}$$

This equation determines the required motor voltage at a given angular velocity to produce a target torque, which can be calculated using the inverse dynamics as discussed below.

3.2 Friction Model

The above motor model describes an ideal DC motor. Real robot joints suffer from friction effects not only inside the motor itself but also in connected transmissions. The focus of our research is on gear transmissions, as they are the most common type of transmission used. We will not consider gear inertia, but discuss a way how they could be included.

The torque τ generated by the DC motor is converted into the output torque τ_o and the friction torque τ_f. Motor axis angular velocity ω and joint velocity $\dot{q}$ are tightly coupled by the gear constant:

$$\tau = \tau_o + \tau_f, \tag{6}$$

$$\omega = k_{gear}\dot{q}. \tag{7}$$

The dominant friction forces result from static and Coulomb friction in the gears and bearings. The transition between these friction states is known as the Stribeck curve [7]. A common choice for this transition is an exponential decrease from static to Coulomb friction developed by Bo and Pavelescu [8] and refined by

Armstrong-Helouvry [9] with an additional viscous friction term for lubricated gear teeth:

$$\tau_f = \operatorname{sgn}(\dot{q})\left((1-\beta)\tau_c + \beta\tau_s\right) + c^{(v)}\dot{q}, \tag{8}$$

$$\beta = \exp\left(-\left|\frac{\dot{q}}{\dot{q}^{(s)}}\right|^{\delta}\right), \tag{9}$$

where q is the joint position (angular or linear), τ_c and τ_s describe Coulomb and static friction torques, and $c^{(v)}$ is the viscous friction constant. The Stribeck parameter $\dot{q}^{(s)}$ determines the transition velocity between static and Coulomb friction. The empirical exponent δ depends on the material surfaces and ranges from 0.5 to 1.0 [3].

The combination of motor and friction models results in the equation

$$U = \omega k_\tau + \frac{R}{k_\tau}\left(\tau_o + \operatorname{sgn}(\dot{q})\left((1-\beta)\tau_c + \beta\tau_s\right) + c^{(v)}\dot{q}\right) \tag{10}$$

$$= \dot{q}k_{gear}k_\tau + \frac{R}{k_\tau}\tau_o + \tau_c \operatorname{sgn}(\dot{q})\frac{R}{k_\tau}(1-\beta) + \tau_s \operatorname{sgn}(\dot{q})\frac{R}{k_\tau}\beta + \frac{R}{k_\tau}c^{(v)}\dot{q} \tag{11}$$

$$= \frac{R}{k_\tau}\tau_o + (k_{gear}k_\tau + \frac{R}{k_\tau}c^{(v)})\dot{q} + \tau_c\frac{R}{k_\tau}\operatorname{sgn}(\dot{q})(1-\beta) + \tau_s\frac{R}{k_\tau}\operatorname{sgn}(\dot{q})\beta, \tag{12}$$

which can be simplified to

$$U = \alpha_0\tau_o + \alpha_1\dot{q} + \alpha_2 \operatorname{sgn}(\dot{q})(1-\beta) + \alpha_3 \operatorname{sgn}(\dot{q})\beta. \tag{13}$$

As exact values for the physical motor and friction constants are not required, it suffices to determine the α_i to obtain a motor model usable for control. The α_i are linear coefficients and can be estimated from experimental data using regression techniques.

One should note that β depends on the Stribeck parameters δ and $\dot{q}^{(s)}$, which cannot be estimated using linear optimization. Non-linear optimization techniques may be employed to solve this problem, but have their own shortcomings, as they might find local optima which generate physically incorrect solutions [10].

As the influence of the Stribeck parameters is limited to very low velocities, it is satisfactory in most cases to use a reasonable fixed transition speed and set the exponent δ to 1 in order to simplify the calculations. If low velocities are an important aspect of the robot's operation, hand-tuning of the Stribeck parameters might be necessary.

If explicit consideration of gear inertia is needed, a fifth term needs to be added, as the torque generated by gear inertia is directly proportional to the angular acceleration $\ddot{q}$ of the joint axis:

$$U = \alpha_0\tau_o + \alpha_1\dot{q} + \alpha_2 \operatorname{sgn}(\dot{q})(1-\beta) + \alpha_3 \operatorname{sgn}(\dot{q})\beta + \alpha_4\ddot{q}. \tag{14}$$

In our experiments, consideration of gear inertia was not required for good model performance.

A reaction time t_d needs to be included in the model equation if there are significant time delays in the system:

$$U(t - t_d) = \alpha_0 \tau_o(t) + \alpha_1 \dot{q}(t) + \alpha_2 \operatorname{sgn}(\dot{q}(t))(1 - \beta(t)) + \alpha_3 \operatorname{sgn}(\dot{q}(t))\beta(t). \quad (15)$$

3.3 Adaption to Position-Controlled Actuators

Most of the available intelligent actuators for robotics are position-controlled servo motors. Since the applied motor voltage cannot be directly influenced, a relationship between the command input of the actuator and the applied motor voltage needs to be found. The actuators usually include a complete PID position controller with configurable gains. In most cases, however, only the proportional part P is used, which results in:

$$U = U_{bat} k_{ctrl} k_P (q_d - q), \quad (16)$$

where U_{bat} is the supply voltage, k_P is the configurable P controller gain and k_{ctrl} maps the controller output to motor voltage U. The current and desired servo positions are described by q and q_d, respectively. Since q_d is the command variable, we solve for it:

$$q_d = \frac{1}{k_{ctrl} k_P U_{bat}} U + q \quad (17)$$

$$= \frac{1}{k_{ctrl} k_P U_{bat}} (\alpha_0 \tau_o + \alpha_1 \dot{q} + \alpha_2 (1 - \beta) + \alpha_3 \beta) + q. \quad (18)$$

As k_{ctrl} is unknown, it is combined with the model coefficients α_i:

$$q_d = \frac{1}{k_P U_{bat}} (\hat{\alpha_0} \tau_o + \hat{\alpha_1} \dot{q} + \hat{\alpha_2} (1 - \beta) + \hat{\alpha_3} \beta) + q. \quad (19)$$

The proposed learning and identification procedure can then be performed as presented below. Care should be taken to select a small enough k_P. If k_P is big, the model influence will be small, resulting in bad model fit precision.

The explicit consideration of the supply voltage U_{bat} compensates drops in voltage due to battery draining and ensures good model match over wide voltage ranges.

One should note that the feedback characteristic of the P controller is not modified as long as the model is correct. This means positional errors lead to proportional responses in voltage, just offset by the voltage required to generate τ_o.

4 System Identification

4.1 Reference Trajectory and Learning Process

During the learning process, the robot repeatedly executes a fixed reference trajectory function $q_{ref}(t)$ on a single joint, which in turn defines the goal servo

position over time. The reference trajectory should cover a wide range of servo motions, especially those relevant in later operation. A reference trajectory for a soccer robot should, for instance, include walking and kicking motions.

The first and second derivative of the reference trajectory are needed, since both appear in the motor and friction models and/or dynamics equations which are required to calculate τ_o.

We apply the classic Iterative Learning Control (ILC) algorithm introduced by Arimoto et al. [1] to determine the command inputs from one iteration k to the next iteration $k+1$:

$$U^{(0)}(t) = 0, \tag{20}$$

$$U^{(k+1)}(t) = U^{(k)}(t) + \lambda(e^{(k)}(t + t_d)), \tag{21}$$

where λ is the learning feedback coefficient and $e^{(k)}$ denotes the trajectory error in the k-th iteration. The first command curve $U^{(0)}(t)$ can also be initialized with a guessed command curve (e.g. using the output of an existing controller) to make the algorithm converge faster.

4.2 Parameter Estimation

When the trajectory error after K ILC iterations is sufficiently small, the combined motor and friction model can be fitted against the command curve $U^{(K)}$ generated with ILC. A discrete sampling with N samples is used, while simultaneously compensating the system reaction time:

$$\hat{U}(n) = U^{(K)}(n\Delta t), \tag{22}$$

$$\hat{\tau}_o(n) = \tau_o(n\Delta t + t_d), \tag{23}$$

$$\hat{q}(n) = q(n\Delta t + t_d), \tag{24}$$

$$\hat{\beta}(n) = \beta(n\Delta t + t_d). \tag{25}$$

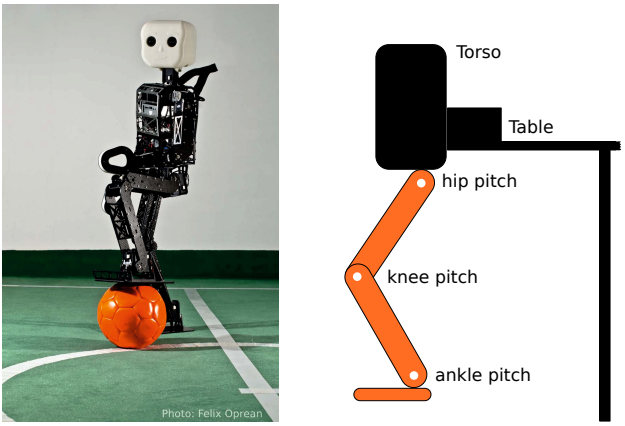
The parameter identification can then be modeled as a least-squares linear optimization problem:

$$\begin{pmatrix} \alpha_0 \\ \alpha_1 \\ \alpha_2 \\ \alpha_3 \end{pmatrix} = \arg\min \left\| A \begin{pmatrix} \alpha_0 \\ \alpha_1 \\ \alpha_2 \\ \alpha_3 \end{pmatrix} - \begin{pmatrix} \hat{U}(0) \\ \vdots \\ \hat{U}(N-1) \end{pmatrix} \right\|^2 \tag{26}$$

with

$$A = \begin{pmatrix} \hat{\tau}_o(0) & \dot{\hat{q}}(0) & \operatorname{sgn}(\dot{\hat{q}}(0))(1-\hat{\beta}(0)) & \operatorname{sgn}(\dot{\hat{q}}(0))\hat{\beta}(0) \\ \vdots & \vdots & \vdots & \vdots \\ \hat{\tau}_o(N-1) & \dot{\hat{q}}(0) & \operatorname{sgn}(\dot{\hat{q}}(N-1))(1-\hat{\beta}(N-1)) & \operatorname{sgn}(\dot{\hat{q}}(N-1))\hat{\beta}(N-1) \end{pmatrix}. \tag{27}$$

Traditional methods including SVD or QR factorization can be used to solve for α_i.



(a) (b)

Fig. 1. Experimental setup: (a) NimbRo-OP prototype on which the experiments were carried out; (b) Setup for single joint evaluations. Fixed robot parts are drawn in *black*, while the *orange* leg can move freely.

5 Experimental Results

To validate our approach, we performed experiments on the NimbRo-OP Humanoid TeenSize Open Platform robot [2] (see Fig. 1(a)). NimbRo-OP primarily uses Dynamixel MX-106 intelligent actuators from Robotis Inc. in its legs.

We used a newly introduced inverse dynamics module based on the Recursive Newton-Euler Algorithm [11] contained in the open-source Rigid Body Dynamics Library (RBDL, see [12]) to calculate the torques needed for achieving desired joint trajectories. For these calculations, we created a full kinematic model of our robot, including inertial information. We estimate the direction of gravity using the built-in inertial measurement unit of NimbRo-OP.

5.1 Single Joint Evaluation

For evaluation on the single joint level, we fixed the NimbRo-OP torso to a table and used a hip pitch actuator of the robot (see Fig. 1(b)). The other actuators of the leg were commanded to hold position. Given that the servo is position controlled, Equation (19) models the system. We choose a reference trajectory that contained elements relevant to the robot application (robot soccer), as shown in Fig. 2. We derived velocities and accelerations from the reference trajectory and calculated torques using the inverse dynamics module.

We fixed the Stribeck parameters $\dot{q}^{(s)}$ and δ to initial assumptions and estimated the system reaction time t_d from the time delay between command and reaction under P control (see Fig. 5). We increased the ILC learning coefficient λ slowly until a sufficient convergence speed was reached. The final parameter values are summarized in Table 1(a).

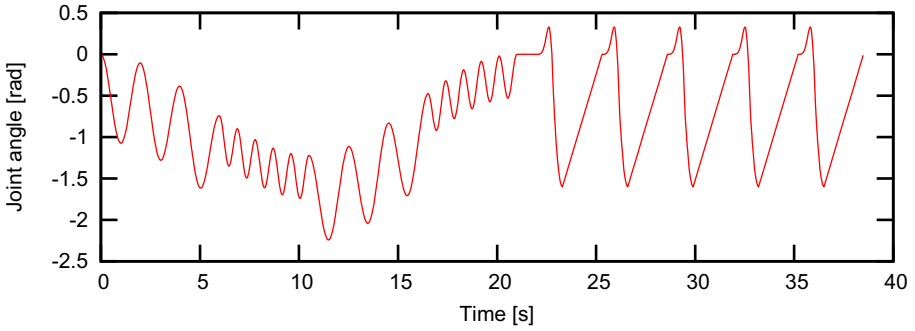


Fig. 2. The reference trajectory used in the experiments. The first section contains sinusoidal oscillations as they occur during walking motions. The second section is composed of fast sinusoidal and linear parts often used for kicking motions.

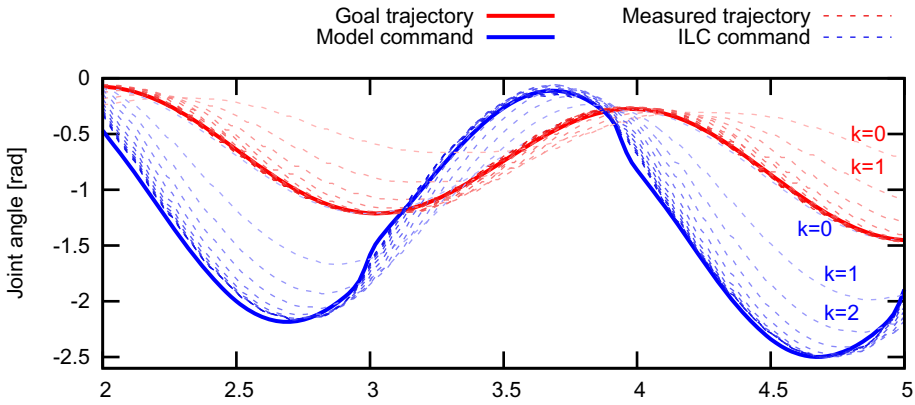


Fig. 3. Learning process applied on the reference trajectory with 12 iterations of ILC. Note that the first ILC iteration ($k = 0$) starts at the goal trajectory, with further iterations converging towards the model command curve fitted in the 12th iteration.

The learning process is visualized in Fig. 3. During the learning process, the trajectory deviation was measured in a windowed fashion to allow for latencies:

$$q_{max} = \max_{t} \min_{-D<a<D} |q_{ref}(t) - q_m(t+a)|\,, \tag{28}$$

where $q_m(t)$ denotes the measured trajectory.

A total of 12 ILC iterations were needed to produce an acceptable maximum trajectory deviation of approx. 0.02 rad for a window size $D = 25$ ms. Note that the learned commands sent to the motor often significantly deviate from the reference trajectory by more than 1 rad. The model was then fitted using least squares. The fitted model produced a maximum trajectory deviation of 0.1 rad on the reference trajectory. Fig. 4 shows the detailed development of the trajectory error. The identified parameters are summarized in Table 1(b).

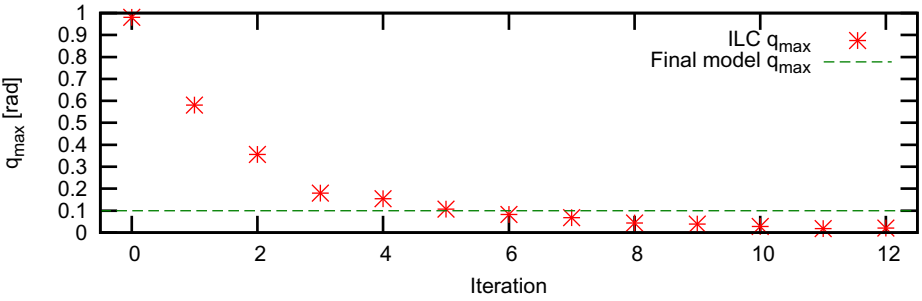


Fig. 4. Maximum windowed trajectory error q_{max} (see Eq. 28) during the learning process

Table 1. (a) Chosen parameters of the learning process; (b) identified model parameters for ROBOTIS Dynamixel MX-106

$\dot{q}^{(s)}$	δ	t_d	λ
0.1 rad/s	1.0	0.03 s	0.8

(a)

$\hat{\alpha}_0$	$\hat{\alpha}_1$	$\hat{\alpha}_2$	$\hat{\alpha}_3$
0.19817	0.49586	0.13729	0.03006

(b)

5.2 Integration into a Walking Motion

We integrated the resulting motor model into the emerging NimbRo-OP soccer software framework [13]. We used the model for feed-forward control only and did not incorporate feedback mechanisms in order to avoid oscillations, which could be caused by latencies, and to keep the robot compliant to outside disturbances.

Since our robot currently does not provide a way of measuring foot contact forces, we move the origin of the inverse dynamics calculation to the support foot under the assumption that it is essentially fixed to the ground (i.e. does not slide or tilt). In this way, the calculation does not depend on the contact forces. During support transition, when both feet are on the ground, the estimated torque is faded in a linear fashion between the computed results of the inverse dynamics for both feet.

The actuators of previous NimbRo robots [14] had to be driven with relatively high proportional gains to meet trajectory requirements. The new feed-forward control based on the learned motor model allows very low P gains to be used while still following the target trajectory (see Fig. 5). In comparison to hard, high-gain P control, the model-based control uses less power: Energy consumption was measured with the internal current sensor of the actuator. After 40 full steps, the left knee joint had consumed 189 J. The model-based control resulted in 140 J of consumed energy. Two factors contribute to this reduction of energy usage. Current spikes are avoided by predicting higher loads, as can be seen in Fig. 5. Steady-state current is also reduced. Less current consumption leads to longer battery life and less heat in the actuators—both positive effects.

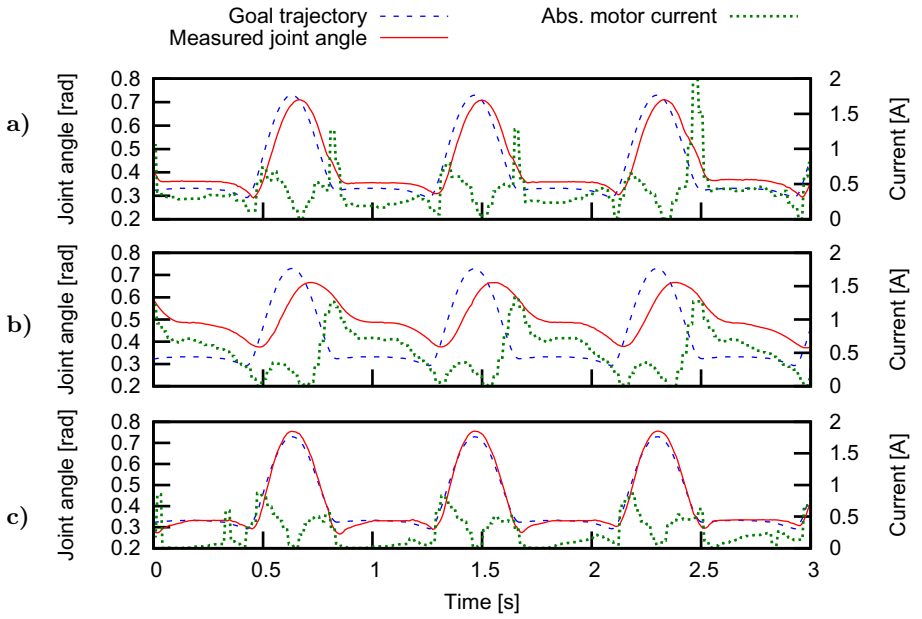
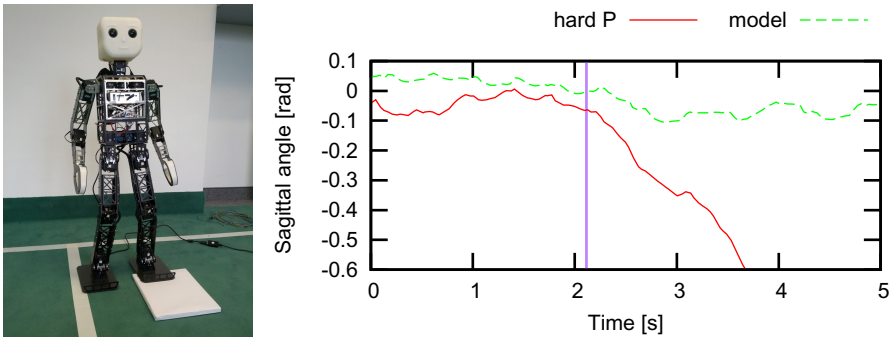


Fig. 5. Evaluation of feed-forward control with the learned joint model on a full-body walking motion. Shown are the joint targets and measured angles of the left knee. The leg contracts with a positive knee angle. **a)** Trajectory tracking using high-gain P control ($k_p = 1.0$). **b)** Trajectory tracking using low-gain P control ($k_p = 0.35$). **c)** Trajectory tracking using low-gain P control ($k_p = 0.35$) with model-based feed-forward control.



(a) (b)

Fig. 6. Obstacle experiment. (a) Experiment setup. The obstacle has a height of 15 mm; (b) Development of sagittal trunk angle during the experiment. The *purple* line denotes the time of impact of the first step on the obstacle. The applied torque by the hard P controller leads to the robot falling backwards, while the model-based controller reacts softly enough to maintain stability.

As current is directly coupled with torque produced by the motor (see Eq. 4), potentially damaging torque spikes are also avoided.

The incorporated knowledge about the system also results in less latency than the high-gain controller. Furthermore, the explicit modeling of battery voltage in Eq. 19 results in equal performance over the whole battery voltage range.

In the walking experiments, trajectory errors were mainly caused by erroneous foot contact estimation. For example, undesired behavior could be observed when the robot mistakenly thought that one foot is firmly on the ground and moved the base of the dynamics calculations to this foot. If the foot is still in the air, the applied torques caused strange behavior. This process can be seen in the overshoot seen in Fig. 5 on the downward slope. However, since a low-gain P controller is used, the legs are very compliant and move back into proper position as soon as ground contact is established as can be seen in Fig. 5. Hence, walking performance is not adversely affected.

On the contrary, the higher compliance helps NimbRo-OP to walk over small obstacles (see Fig. 6(a)) without stabilizing algorithms. We tested rectangular obstacles with a height of up to 15 mm and commanded the robot to walk forward with a very slow velocity. Fig. 6(a) illustrates the robot behavior when executing the first step onto the obstacle. One can observe that the hard P controller reacts with too much force and causes the robot to fall backwards while the model-based controller is more compliant and just leads to a slight backward leaning. An additional stabilizing factor in this situations is the fact that calculated joint torques reflect the orientation of the robot, as the direction of gravity is estimated by the IMU.

6 Conclusion

In this paper, we demonstrated that simple motor and friction models can be learned using Iterative Learning Control in a fast and straightforward way, avoiding dedicated test runs for single parameters but instead fitting all parameters at once. The learned model performs better than traditional P controllers on real walking motions. Even intelligent servo actuators like the Dynamixel MX series can profit from an accurate motor model for feed-forward control, resulting in less power consumption and more compliant robot movement. Due to the generality of the used DC motor model, and the single assumption of a low-gain position-based P controller, our approach is applicable to a wide range of robotic actuators.

Some of the simplifications made when modeling motor and friction (e.g. unmodelled gear inertia) may be improved in future work. Since friction is the dominant problem in robot joint control, it may for example be prudent to investigate more complex friction models like the more general robot joint friction model proposed by Waiboer [10].

If the model is used in feed-forward control of position-controlled servos, wrongly estimated model parameters or an incorrect dynamics model can result in steady-state trajectory errors which are not corrected. Further work may

include the careful application of feedback mechanisms (e.g. online model refinement) to reduce these errors.

Acknowledgment. This work has been partially funded by grant BE 2556/6 of German Research Foundation (DFG).

References

1. Arimoto, S., Kawamura, S., Miyazaki, F.: Bettering operation of robots by learning. Journal of Robotic Systems 1(2), 123–140 (1984)
2. Schwarz, M., Schreiber, M., Schueller, S., Missura, M., Behnke, S.: NimbRo-OP humanoid TeenSize open platform. In: 7th Workshop on Humanoid Soccer Robots, IEEE-RAS International Conference on Humanoid Robots, Osaka (2012)
3. Waiboer, R., Aarts, R., Jonker, B.: Velocity dependence of joint friction in robotic manipulators with gear transmissions. In: ECCOMAS Thematic Conference Multibody Dynamics (2005)
4. Mensink, A.: Characterization and modeling of a dynamixel servo. Technical report, University of Twente, Individual research assignment (2008)
5. Kim, Y.H., Lewis, F.L.: Reinforcement adaptive learning neural-net-based friction compensation control for high speed and precision. IEEE Transactions on Control Systems Technology 8(1), 118–126 (2000)
6. Liu, J.-S.: Joint stick-slip friction compensation for robotic manipulators by iterative learning. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), vol. 1, pp. 502–509 (1994)
7. Stribeck, R., Schröter, M.: Die wesentlichen Eigenschaften der Gleit- und Rollenlager: Untersuchung einer Tandem-Verbundmaschine von 1000 PS. Springer (1903)
8. Chun Bo, L., Pavelescu, D.: The friction-speed relation and its influence on the critical velocity of stick-slip motion. Wear 82(3), 277–289 (1982)
9. Armstrong-Helouvry, B.: Control of Machines with Friction. International Series in Engineering and Computer Science, vol. 128. Springer (1991)
10. Waiboer, R.R.: Dynamic Modelling, Identification and Simulation of Industrial Robots – for off-line programming of robotised laser welding. University of Twente (2007)
11. Featherstone, R.: Rigid Body Dynamics Algorithms. Springer, Berlin (2008)
12. Rigid Body Dynamics Library, `https://bitbucket.org/rbdl/rbdl/`
13. Pastrana, J., Schwarz, M., Schreiber, M., Schueller, S., Missura, M., Behnke, S.: Humanoid TeenSize open platform NimbRo-OP. In: Proceedings of 17th International RoboCup Symposium (2013)
14. Missura, M., Münstermann, C., Mauelshagen, M., Schreiber, M., Behnke, S.: RoboCup 2012 Best Humanoid Award winner NimbRo TeenSize. In: Chen, X., Stone, P., Sucar, L.E., van der Zant, T. (eds.) RoboCup 2012. LNCS (LNAI), vol. 7500, pp. 89–93. Springer, Heidelberg (2013)