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

Learning to Regulate Rolling Ball Motion

Jha, Devesh K.; Yerazunis, William S.; Nikovski, Daniel N.; Farahmand, Amir-massoud

TR2017-176 November 27, 2017

Abstract

In this paper, we present a problem of regulating the motion of a rolling ball in a one-dimensional space in the presence of non-linear effects of friction and contact. The regulation problem is solved using a model-based reinforcement learning technique. A Gaussian process model is learned to make predictions on the motion of the ball and then, the predictive model is used to solve for the control policy using dynamic programming by estimating the value functions. Several results are shown to demonstrate the simple, yet interesting motion dynamics for the ball. Our hope is that the proposed system will serve as a simple benchmark system for reinforcement and robot learning.

IEEE Symposium on Computational Intelligence in Engineering Solutions

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

Learning to Regulate Rolling Ball Motion

Devesh K. Jha
Mitsubishi Electric Research Laboratories
Cambridge, MA USA
jha@merl.com

William Yerazunis
Mitsubishi Electric Research Laboratories
Cambridge, MA USA
yerazunis@merl.com

Daniel Nikovski
Mitsubishi Electric Research Laboratories
Cambridge, MA USA
nikovski@merl.com

Amir-massoud Farahmand
Mitsubishi Electric Research Laboratories
Cambridge, MA USA
farahmand@merl.com

***Abstract*—In this paper, we present a problem of regulating the motion of a rolling ball in a one-dimensional space in the presence of non-linear effects of friction and contact. The regulation problem is solved using a model-based reinforcement learning technique. A Gaussian process model is learned to make predictions on the motion of the ball and then, the predictive model is used to solve for the control policy using dynamic programming by estimating the value functions. Several results are shown to demonstrate the simple, yet interesting motion dynamics for the ball. Our hope is that the proposed system will serve as a simple benchmark system for reinforcement and robot learning.**

***Index Terms*—Reinforcement Learning, Gaussian Process, Position Regulation**

I. Introduction

This paper presents a study of regulating the position of a rolling ball in a one-dimensional space using reinforcement learning [18]. The problem is motivated by a general class of systems with contact dynamics and various other kinds of non-linearities like dry friction and hysteresis [9]. To synthesize accurate controllers for such systems, one needs to account for the above-mentioned non-linear effects on system dynamics. Making physical models for these systems which can then be used for their high precision control is generally very difficult and requires lot of domain knowledge for precise modeling. Reinforcement learning [18], on the other hand, provides a data-driven approach to control these non-linear systems without creating detailed physical models for the same and is thus an attractive alternative.

Reinforcement learning algorithms can broadly be classified in two categories [12], [18]– model-based and model-free reinforcement learning. Model-Based Reinforcement learning (MBRL) techniques use examples to learn a model of the environment, which is then used to find a policy for controlling the system ([5], [10], [14]). The notion of model-free reinforcement learning is to bypass the associated model learning and rather learn the policy using a value or policy gradient algorithm directly from the data ([1], [6], [17]). While some of the classical model-free approaches like Q-learning, TD-learning [18], etc. have nice theoretical asymptotic convergence guarantees, they require a large number of samples for convergence and thus could be an impractical option for robotic systems where exploration could be prohibitively costly. The model-based reinforcement learning techniques are, in general, more data efficient than the model-free approaches [3], [5]. However, the performance of these approaches depend on the accuracy of the learned model, and any imperfection in the model propagates into the synthesized policy. A good survey of reinforcement learning techniques for robotics could be found in [12]. Gaussian process (GP) models have attracted a lot of attention in reinforcement learning community [7], [8], [11]. A lot of work has been done recently to exploit uncertainties in the predictive models for policy synthesis [4], [5]. Our work is most closely related to the work presented in [7] on Gaussian process dynamic programming.

In this paper, we use a model-based reinforcement learning technique to solve a regulation problem for a one-dimensional system. The system consists of a metallic ball that can roll along the length of a bar mounted on a tip-tilt platform. The task is to regulate the position of the ball at a specified location by controlling the tip angle of the table. This is done by first learning a predictive model for the motion of the ball using Gaussian processes. The Gaussian process model is then used to synthesize a policy by discretizing the system state and input space. Performance of the MBRL policy is compared to a PID controller. We show that the learned policy is able to achieve bang-bang type of behavior for regulating the position of the system.

Contributions. The paper has the following contributions:

1) We present a low-dimensional control problem which can possibly be used as a benchmark for reinforcement (and robot) learning problems.
2) We use a model-based reinforcement learning technique to solve the regulation problem and present the related analysis.

In Section II, we present a brief overview of Gaussian processes for the completeness of the paper. Section III presents a simplified simulated example as well as the observed system

behavior. In Section IV, we present the model-based reinforcement learning solution to the regulation problem with some comparisons to a conventional PID controller. Finally, the paper is summarized and concluded with some future research directions in Section V.

II. Background

In this section, we provide a background on Gaussian processes which is used to learn the probabilistic model for the motion of the presented system. Interested readers are referred to [16] for detailed description and analysis of Gaussian processes. Using Gaussian processes for model learning is motivated by the fact that it allows computation of the uncertainties associated with system dynamics which can be used to solve the related Markov Decision Problem (MDP) for reinforcement learning more efficiently due a better model.

Definition 2.1 (Gaussian Processes): A Gaussian process is a collection of random variables, any finite number of which have a joint Gaussian distribution.

A Gaussian process (GP) is completely specified by its mean and covariance function. We define the mean function $m(\mathbf{x})$ and covariance function $k(\mathbf{x}, \mathbf{x}')$ of a real process $f(\mathbf{x})$ as follows.

$$m(\mathbf{x}) = \mathbb{E}[f(\mathbf{x})]$$
$$k(\mathbf{x}, \mathbf{x}') = \mathbb{E}[(f(\mathbf{x}) - m(\mathbf{x}))(f(\mathbf{x}') - m(\mathbf{x}'))]$$

and the Gaussian process is written as follows.

$$f(\mathbf{x}) \sim \mathcal{GP}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')) \tag{1}$$

A Gaussian process is defined as a collection of random variables which means that the if the GP specifies $(x_1, x_2) \sim \mathcal{N}(\mu, \Sigma)$, then it must also specify $x_1 \sim \mathcal{N}(\mu_1, \Sigma_{11})$ where Σ_{11} is the relevant submatrix of Σ. Some of the common choice for GP covariance function are rational quadratic (RQ), squared exponential (SE), Matérn covariance functions etc. [16]. Some of the functional forms of the covariance function and the associated free parameters are listed in Table I. The inference problem associated with GP is to infer the related hyperparameters (or the free parameters). The associated hyperparameters with a GP are the parameters corresponding to the mean and covariance function for the GP. The inference methods compute an approximate posterior, an approximate negative log marginal likelihood and its partial derivatives w.r.t. the hyperparameters, given the choice of mean, covariance and likelihood functions. Some of the common inference methods are expectation propagation, Laplace's approximation, Kullback-Leibler (K-L) divergence minimisation, etc. [16].

TABLE I: Some examples of covariance functions used for Gaussian Process. The first and second are Squared Exponential (SE) and the third function is a rational quadratic (RQ).

$k(\mathbf{x}, \mathbf{x}')$	Free Parameters
$\sigma^2 \exp(-(x - x')^T \Lambda^{-1} (x - x')/2)$	$\{\lambda_1, \ldots, \lambda_D, \sigma\}$
$\sigma^2 \exp(-(x - x')^T (x - x')/2\ell^2)$	$\{\ell, \sigma\}$
$\sigma^2 (1 + \frac{1}{2\alpha}(x - x')^T \Lambda^{-2} (x - x'))^{-\alpha}$	$\{\lambda_1, \ldots, \lambda_D, \sigma, \alpha\}$

III. System Dynamics

In this section we present an overview of the experimental system and present some simplistic analysis of its dynamics. Figure 1 shows the experimental setup with the tip-tilt-rotate table. An aluminum bar is attached to the tip-tilt-rotate table (platform) with three degrees of freedom. The platform has three inexpensive, off-the-shelf HiTec type HS-805BB RC model PWM-controlled servo motors that provide accurate open-loop positioning but do not provide any feedback of the position. The aluminum bar is approximately $22\,\mathrm{cm}$ long and has a groove along its length. We use a brass ball of diameter approximately $1.6\,\mathrm{cm}$. The aluminium bar groove is an as-acquired extrusion, and not particularly precise; it has several rough areas on the order of 100 microns deviation from flat. Likewise, the brass ball is painted with an orange fluorescent paint that adheres only poorly to the surface, with several bare spots. This causes the ball-and-track system to have a significant nonlinear response, with significant stop-start dry-friction hysteresis in some configurations but not others. For controlling the motion of the ball on the bar, we use only one of the three degrees-of-freedom of the table to control the slope of the bar. The bar has steel bolts at the ends to keep the ball from jumping off the bar. While this arrangement is required to constrain the ball on the bar, it leads to discontinuity in the motion dynamics at the two ends (and thus can lead to modeling errors). We use an RGB camera which is attached to a fixed frame to estimate the states of the system. The ball is tracked in real-time using a simple, yet fast, blob tracking algorithm along with a Kalman filter. All the communication with the camera and servo motors driving the system is done using the Robot Operating System (ROS) [15].

To gain an insight on the system dynamics, we analyze a simplified system in simulation. We model the system as a point mass with constant mass and a constant coefficient of kinetic friction for the one-dimensional space. The equations of motion of the idealized system can be written as follows:

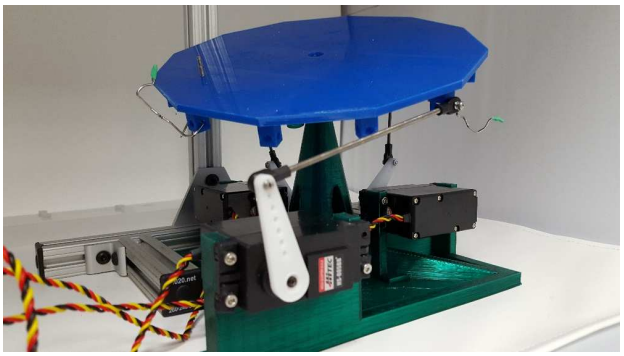
$$\dot{x}_1 = x_2 \tag{2}$$
$$\dot{x}_2 = -cx_2 + u$$

where x_1, x_2 are position and velocity of the system, c is the coefficient of kinetic friction of the system and u is the control input (for our system it would be the $\sin(\phi)$, where ϕ represents the tip angle of the bar). In matrix form, it can be be written as follows:

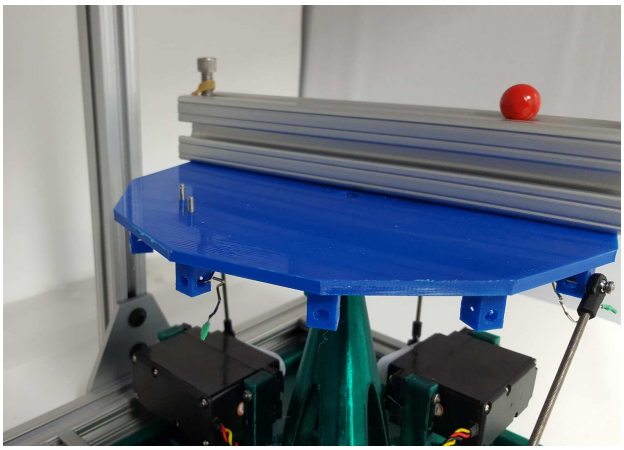
$$\dot{\mathbf{x}} = A\mathbf{x} + Bu$$

where $A = \begin{bmatrix} 0 & 1 \\ 0 & -c \end{bmatrix}$ and $B = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$.

It is noted that the model given by Equation 2 treats the ball as a point mass and ignores any rotational motion it has and assumes that the coefficient of friction is a constant for all points of contact on the bar. In Figure 2, we show



(a) The tip-tilt-rotate platform with three servo motors



(b) The one dimensional set up mounted on the tip-tilt-rotate platform

Fig. 1: The experimental setup used in the paper

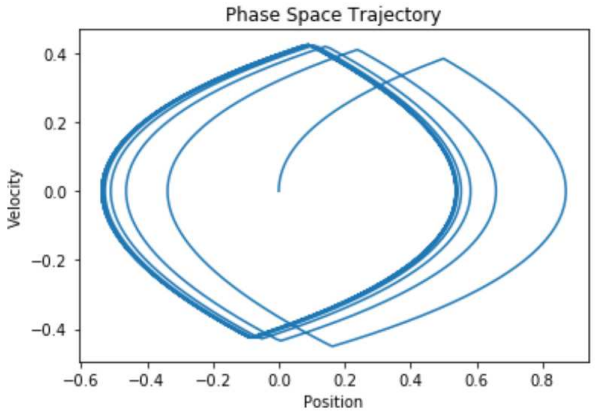


Fig. 2: This plot shows the phase plot of the simplified system when excited with a sine wave. The observed behavior shows a periodic motion for the system.

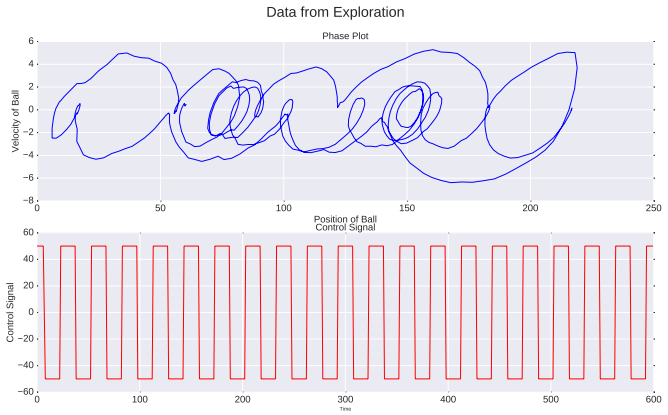


Fig. 3: This plot shows the phase plot of the actual system when excited with a square wave excitation. This is very different from the expected simulated behavior.

simulated trajectories for the simplistic model with a sine wave excitation. The phase-space of the system shows a periodic motion. Another important point to note here that for the simplified point-mass object, the optimal controller for a regulation problem could be proven to be bang-bang using the bang-bang principle for linear system [2].

In Figure 3, we show the phase space of the system when excited with a square wave. The frequency was selected so as to minimize the chances of the ball hitting the end bolts and thus to keep the ball oscillating between them. The results as seen in Figure 3 are quite different from the expected behavior of the point particle in the simulated examples. The system depicts a highly non-linear behavior and the system behavior doesn't converge to a limit cycle as was expected from the simulated results. This difference could be mainly attributed to the non-uniform static friction coefficient, the point of contact of the ball and the rolling motion of the ball itself. These effects were not considered in the simulated model as they are very difficult to model in general. As a result of friction and the rotational motion of the ball, the ball can't have a single stable limit cycle (as seen in Figure 3).

IV. Proposed Approach

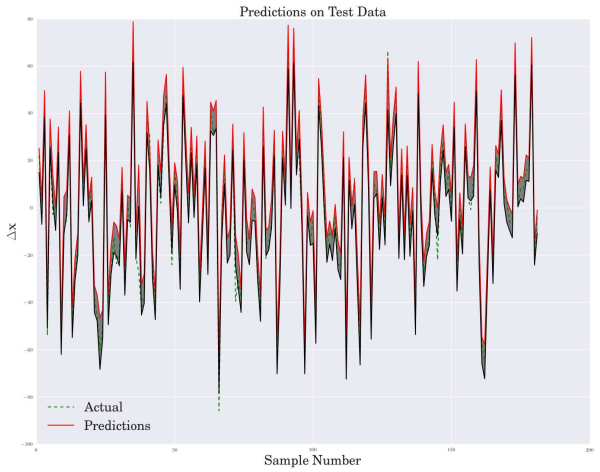
In this section, we present analysis of the regulation problem. We first present model learning using Gaussian processes and then present details of policy synthesis.

A. Model Learning

In this section, we present analysis for learning a predictive model for the system and detail related results. We learn a probabilistic model for the system, that predicts the expected change in the position and velocity of the system given the current position, velocity and the control input. More concretely, we learn a predictive model of a dynamical system as follows:

$$\mathbf{x}_t = f(\mathbf{x}_{t-1}, \mathbf{u}_{t-1}) \tag{3}$$

where $\mathbf{x}_t \in \mathbb{R}^N$ denotes the state vector at time t and $\mathbf{u}_t \in \mathbb{R}^M$ denotes the input vector at time t. The function f is an unknown function that we want to estimate from the training data. We use the tuples $(\mathbf{x}_t, \mathbf{u}_t)$ as the training inputs and the differences $\Delta_t = \mathbf{x}_t - \mathbf{x}_{t-1}$ as the training targets. This input-output relationship is modeled as a GP and denoted by the function f. Using the training data and the corresponding training targets, we learn the corresponding



(a) GP predictions of position with the corresponding uncertainty band(in mm)

(b) GP predictions of velocity with the corresponding uncertainty band (in $\mathrm{mm\,s^{-1}}$)

Fig. 4: Result of GP predictions on test data .

GP hyperparameters (see section II). In our work, we have used the Matern covariance (with parameter $\nu = 2.5$) function as the choice of the kernel function. The hyperparameters of the GP model are estimated by maximizing the log marginal likelihood using the scikit-learn package. The GP-posterior is then a one-step prediction model. Under assumptions of GP modeling, the predictions are Gaussian distributed which are shown in the following equations.

$$p(x_t|x_{t-1}, u_{t-1}) = \mathcal{N}(x_t|\mu_t, \Sigma_t) \tag{4}$$

$$\mu_t = x_t + \mathbb{E}_f[\Delta_t] \tag{5}$$

$$\Sigma_t = \mathrm{var}_f[\Delta_t] \tag{6}$$

For multi-dimensional targets, we train conditionally independent GP models for each target dimension.

In Figure 4, we show the result of GP predictions along with the uncertainty bands (corresponding to a significance level of 95%) for position and velocity respectively. The results are also shown and compared with random forests and ridge regression (as a representative of a linear model) in Table II. For GP models, the errors are calculated based on the mean predictions. In other words, we donot use the associated uncertainty in the GP model in planning. This is left as an exercise for future work. Overall, it is seen that GP performs better than the linear ridge regression as well the random forest regression.

TABLE II: Training error results for different models. Lower is better.

Model	Position (mm)		Velocity ($\mathrm{mm\,s^{-1}}$)	
	Training	Test	Training	Test
Gaussian Process	2.2	3.5	0.67	0.76
Random Forest	1.76	4.86	0.67	1.00
Ridge Regression	5.14	5.73	1.76	1.64

B. Policy Synthesis

In this section, we present the details of the policy synthesis steps using the learned model from section IV-A. The objective is to find a deterministic policy/controller $\pi : \mathbf{x} \mapsto \pi(\mathbf{x}) = \mathbf{u}(x)$ that minimizes the expected cost which is given by the following equation:

$$J_\theta(\mathbf{x}) = \min_{a \in A}\{c(\mathbf{x}, a) + \gamma J_\theta(\hat{\mathbf{x}})\}, \tag{7}$$

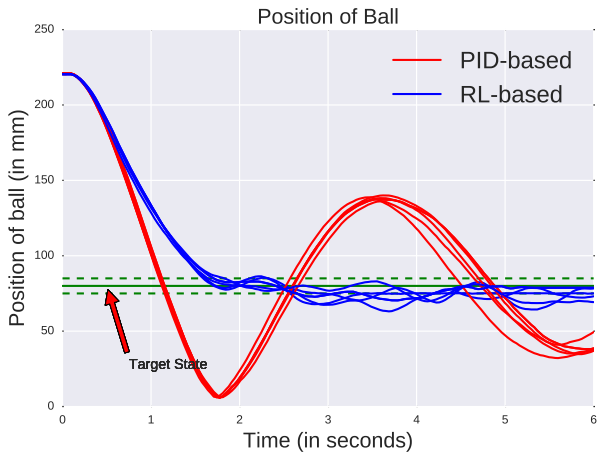
where J_θ is the value function parameterized by θ and γ is the discount factor. In our policy-synthesis setup, we use a cost-function that solely penalizes the distance of the current state from the target state. This is, in general, sufficient for solving a regulation problem as reaching a target state with high speeds leads to high long-term costs due to overshooting and thus poor performance. We use a generalized binary cost function given by the following equation [5].

$$c(\mathbf{x}) = 1 - \exp\left\{\frac{-(\mathbf{x}_{\mathtt{target}} - \mathbf{x})^2}{2\sigma^2}\right\} \tag{8}$$

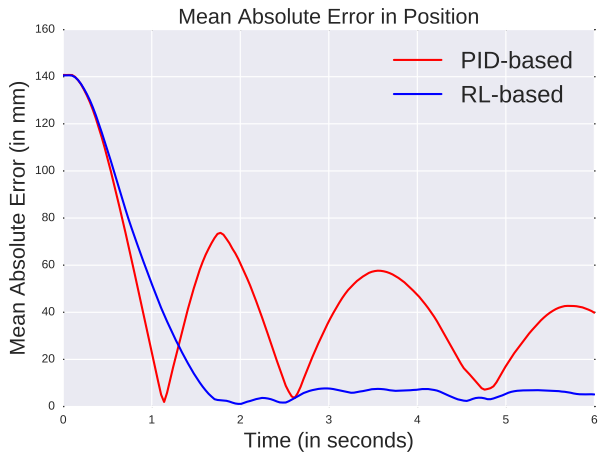
The cost function in Equation (8) acts as a locally quadratic cost function that saturates to one in regions away from the goal state for the system. The local region width is controlled by the parameter σ. In our experiments, we use a σ equal to $5\,\mathrm{mm}$. This in turn implies that the synthesized policy should try to maintain the position of the ball within $5\,\mathrm{mm}$ from the target state.

We synthesize the policy for the regulation problem using value iterations by estimating the value functions. The value functions are estimated on a discrete state-space. The state-space of the system is discretized to a grid of size 200×21, where the position space is discretized into 200 points and the velocity in 21 points. The input space of the system is discretized to 21 steps. The synthesized policy is implemented using a nearest neighbor interpolation in the continuous state-space. The control is implemented at a frequency of $30\,\mathrm{Hz}$.

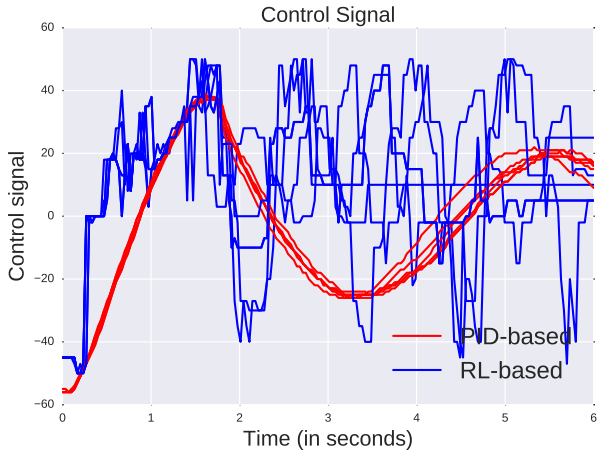
Figure 5 shows the experimental results of the reinforcement learning controller and its comparison to the PID controller.



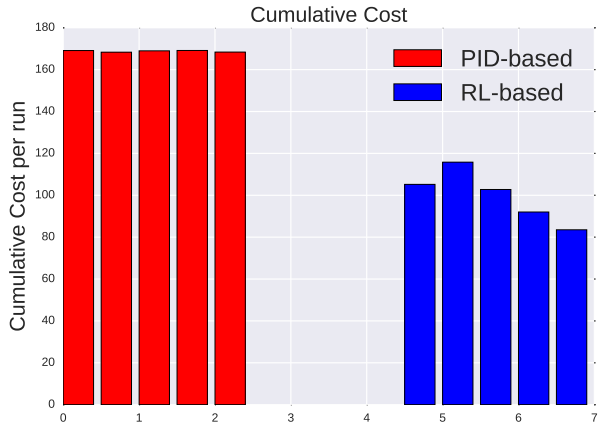
(a) Position of the ball with time for the RL-based and PID controller.



(b) Mean absolute error in the position of the ball from the goal position with time.



(c) Control inputs for the RL-based and PID controller



(d) Cumulative reward for the RL-based and PID controller

Fig. 5: Comparison of the RL-based controller with the PID controller.

The goal of the comparison is to show the advantage of RL-based controller against a controller which doesn't have access to a predictive model (and thus is reactive). The PID controller is implemented with fixed gains. To calculate the statistics of performance of the two controllers, we conduct five independent runs of both controllers from the same initial state of the system. In Figure 5a, we show the plot of position of the ball with time as well the target state for regulation. In Figure 5b, we plot the average absolute error in ball position from the target state. These two plots show the superior performance of the RL controller, and also illustrate that the RL controller is able to maintain the position within $5\,\mathrm{mm}$ of the target state (which was the parameter σ specified in the cost function of). Moreover it shows the planning of the RL controller where the ball first accelerates and then tries to come to rest as it approaches the target state (thus minimizing overshoot). This behavior is similar to the bang-bang control for the simplified point-mass system presented in Section III. The PID controller, however, acts in a reactive fashion and thus incurs higher settling time and steady-state error. The plot in Figure 5a show large oscillations around the target point for the PID controller– such oscillations could be reduced for a system by reducing the controller gain. However, due to the presence of dry friction, we can't reduce the controller gain in our case as the ball doesn't move at times if the tip angle is small depending on the point of contact of the ball with the bar. The difference in behavior of the two controllers could be seen in Figure 5a where the RL-based controller starts decelerating (notice the point after which position-time behavior changes starting from the same state and the RL-based controller moves slower) so as to reach the target state with minimum overshoot. The RL-based controller also suffers from some small oscillations in the neighborhood of the target state. This could be mainly attributed to the modeling error and the state estimation errors.

Figure 5c shows the control signal behavior for the RL and

PID controller where also the difference in the behavior of the two controllers could be noticed. The RL-based controller tries to slow after an initial start (notice the change in sign of the control input). Figure 5d shows the cumulative cost for the two controllers where the RL-based controller performs significantly better. The cumulative cost is calculated for a window of 6 seconds where the cost of an individual state is given by Equation (8). The cost achieved by the RL-controller is much lower than the PID controller. For clarity of presentation, comparisons of the two controllers are listed as a table in Table III.

TABLE III: Performance of RL-based and PID controller. All statistics are calculated at the end of 6 s. Lower is better for all variables.

Controller Type	Mean Error (mm)	Cumulative costs
RL-based	4.8	97.5
PID	40.2	167.5

Even though the RL-based policy shows a bang-bang type behavior, it doesn't seem to be exactly the time-optimal bang-bang solution. Based on the current understanding, this is mainly attributed to the inaccuracies in model learning.

V. Discussions and Summary

In this paper, we presented a one-dimensional motion regulation problem which was solved using a model-based reinforcement learning technique and compared with classical model-free PID controller. Even though the system is constrained to move in a single dimension, it has rich non-linear dynamics for the presence of contact forces, rolling motion and dry friction. We showed that these factors introduce non-linear characteristics in system dynamics, which we believe would be difficult to model from first principles of physics. We learned predictive models from system data using Gaussian processes to predict system states and then, synthesized system policy on a discrete state-space. The time-optimal control for a lot of linear system regulation problems is bang-bang, under certain conditions [2]. From our analysis we showed the learned controller displays bang-bang-type behavior without any knowledge of underlying system dynamics. Our hope is that we can use the insights from this study to analyze more complicated motion models in more complex and constrained environments where friction forces and contact dynamics cannot be ignored.

In the future we would like to learn a better dynamic model for the presented system by explicitly incorporating discontinuities due to contact forces to improve modeling accuracy. We would also like to implement the PILCO framework [5] to compare the benefits it can provide over the current policy synthesis process by making use of the uncertainties in model learning. Another interesting future work is to solve the regulation problem in an end-to-end fashion using deep learning approaches [13].

References

[1] A. Antos, C. Szepesvári, and R. Munos, "Fitted Q-iteration in continuous action-space mdps," in *Advances in neural information processing systems*, 2008, pp. 9–16.

[2] P. J. Antsaklis and A. N. Michel, *A linear systems primer*. Birkhäuser Boston, 2007, vol. 1.

[3] C. G. Atkeson and J. C. Santamaria, "A comparison of direct and model-based reinforcement learning," in *Robotics and Automation, 1997. Proceedings., 1997 IEEE International Conference on*, vol. 4. IEEE, 1997, pp. 3557–3564.

[4] M. Deisenroth and C. E. Rasmussen, "Pilco: A model-based and data-efficient approach to policy search," in *Proceedings of the 28th International Conference on machine learning (ICML-11)*, 2011, pp. 465–472.

[5] M. P. Deisenroth, D. Fox, and C. E. Rasmussen, "Gaussian processes for data-efficient learning in robotics and control," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 2, pp. 408–423, 2015.

[6] M. P. Deisenroth, G. Neumann, J. Peters *et al.*, "A survey on policy search for robotics," *Foundations and Trends® in Robotics*, vol. 2, no. 1–2, pp. 1–142, 2013.

[7] M. P. Deisenroth, C. E. Rasmussen, and J. Peters, "Gaussian process dynamic programming," *Neurocomputing*, vol. 72, no. 7, pp. 1508–1524, 2009.

[8] Y. Engel, S. Mannor, and R. Meir, "Reinforcement learning with gaussian processes," in *Proceedings of the 22nd international conference on Machine learning*. ACM, 2005, pp. 201–208.

[9] G. Gilardi and I. Sharf, "Literature survey of contact dynamics modelling," *Mechanism and machine theory*, vol. 37, no. 10, pp. 1213–1239, 2002.

[10] T. Hester and P. Stone, "Generalized model learning for reinforcement learning in factored domains," in *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems-Volume 2*. International Foundation for Autonomous Agents and Multiagent Systems, 2009, pp. 717–724.

[11] J. Ko, D. J. Klein, D. Fox, and D. Haehnel, "Gaussian processes and reinforcement learning for identification and control of an autonomous blimp," in *Robotics and Automation, 2007 IEEE International Conference on*. IEEE, 2007, pp. 742–747.

[12] J. Kober, J. A. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1238–1274, 2013.

[13] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-end training of deep visuomotor policies," *Journal of Machine Learning Research*, vol. 17, no. 39, pp. 1–40, 2016.

[14] S. Levine and V. Koltun, "Guided policy search," in *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, 2013, pp. 1–9.

[15] M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "Ros: an open-source robot operating system," in *ICRA Workshop on Open Source Software*, 2009.

[16] C. E. Rasmussen and C. K. Williams, *Gaussian processes for machine learning*. MIT press Cambridge, 2006, vol. 1.

[17] M. Riedmiller, "Neural fitted Q iteration-first experiences with a data efficient neural reinforcement learning method," in *ECML*, vol. 3720. Springer, pp. 317–328.

[18] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press Cambridge, 1998, vol. 1, no. 1.