Differentiable Physics Models for Real-world Offline Model-based Reinforcement Learning

Michael Lutter*, Johannes Silberbauer* , Joe Watson, Jan Peters
Computer Science Department, Technical University of Darmstadt
{michael, joe, jan}@robot-learning.de

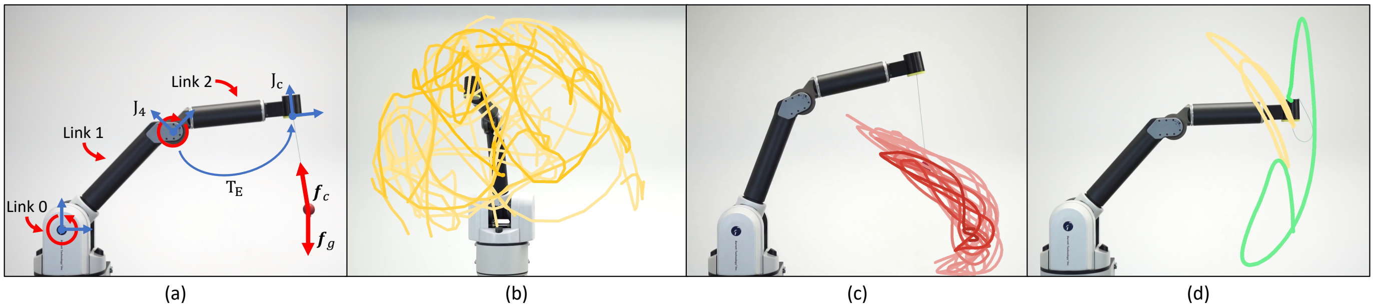

Fig. 1. (a) The identified dynamics (red) and kinematic (blue) parameter of the Barrett WAM for the Ball in a Cup task. (b) Exploration data for the DiffNEA to infer the robot dynamics parameters. (c) Exploration data for the DiffNEA white-box model to infer T_E and the string length. (d) Successful swing-up on real system using offline model based reinforcement learning.

***Abstract*—A limitation of model-based reinforcement learning (MBRL) is the exploitation of errors in the learned models. Black-box models can fit complex dynamics with high fidelity, but their behavior is undefined outside of the data distribution. Physics-based models are better at extrapolating, due to the general validity of their informed structure, but underfit in the real world due to the presence of unmodeled phenomena. In this work, we demonstrate experimentally that for the *offline* model-based reinforcement learning setting, physics-based models can be beneficial compared to high-capacity function approximators if the mechanical structure is known. Physics-based models can learn to perform the ball in a cup (BiC) task on a physical manipulator using only 4 minutes of sampled data using offline MBRL. We find that black-box models consistently produce unviable policies for BiC as all predicted trajectories diverge to physically impossible state, despite having access to more data than the physics-based model. In addition, we generalize the approach of physics parameter identification from modeling holonomic multi-body systems to systems with nonholonomic dynamics using end-to-end automatic differentiation.**

Videos: https://sites.google.com/view/ball-in-a-cup-in-4-minutes/

I. Introduction

The recent advent of model-based reinforcement learning has sparked renewed interest in model learning [1]–[7]. A learned model should reduce the sample complexity of the reinforcement learning task, through interpolation and extrapolation of the acquired data, and thus enable the application to physical systems. Building upon the vast literature of model learning for control, various new approaches to improve black-box models with physics have been proposed. However, the question of what is a good model for MBRL and how this might differ from models for control has not been thoroughly addressed. A popular opinion is that black-box models are preferable, as such models are applicable to arbitrary systems and can approximate complex dynamics with high fidelity. In contrast, physics-based models can underfit due to unmodeled phenomena and require specific domain knowledge about the system.

In this work, we discuss the challenges of model learning for MBRL and contrast them to the challenges of model-based control synthesis, the original motivation for model learning. We compare these requirements to the characteristics of black-box and physics-based models. To experimentally highlight the differences between model representations for MBRL, we compare the performance of each model type using offline MBRL applied to the common RL benchmark of ball in a cup (BiC) [8]–[10] on the physical Barrett WAM. The model performance is evaluated using offline MBRL as this approach is the most susceptible for model exploitation and hence amplifies the differences between model representations. BiC on the Barrett WAM is a challenging task for MBRL as the task requires precise movements, combines various physics phenomena including cable drives, rigid-body-dynamics and string dynamics and uses reduced and maximal coordinates.

In the process we extend the identification of physics models to nonholonomic systems [11], which previously were limited to multi-body kinematic chains [12]–[14]. Using the advancements in automatic differentiation (AD) [15] and careful reparametrizations of the physics parameters one can infer a guaranteed physically plausible model for arbitrary mechanical systems with unconstrained gradient based optimization - if the kinematic structure is known. Thus this extension generalizes

* Equal Contribution

This project has received funding from the European Union's Horizon 2020 program under grant agreement No #640554 (SKILLS4ROBOTS). The research was also supported by grants from ABB and NVIDIA. Calculations were conducted on the Lichtenberg cluster of the TU Darmstadt.

the elegantly crafted features of [12] by backpropagating through the computational graph spanned by the differential equations of physics.

Contributions We provide a experimental evaluation of different model representations for solving BiC with offline MBRL. We show that for some tasks, e.g., BiC, guaranteed physically plausible models are preferable compared to deep networks despite the inherent underfitting. Physics-based white-box models, learned with only four minutes of data, are capable of solving BiC with offline MBRL. Deep network models do not achieve this task. In addition, we extend the existing methods for physics parameters identification to systems with maximal coordinates and nonholonomic inequality constraints.

In the following we discuss the challenges of models for MBRL (Section II), describe our approach to physically plausible parameter identification for systems with holonomic and nonholonomic constraints (Section IV). Finally, Section VI experimentally compares the model representations extensively by applying them to offline MBRL to solve the BiC task on the real Barrett WAM with three different string lengths .

II. Model Representations

Model learning, or system identification [16], aims to infer the parameters $\boldsymbol{\theta}$ of the system dynamics from data containing the system state $\boldsymbol{x}$ and the control signal $\boldsymbol{u}$. In the continuous time case the dynamics are described by

$$\ddot{\boldsymbol{x}} = f(\boldsymbol{x}, \dot{\boldsymbol{x}}, \boldsymbol{u}; \boldsymbol{\theta}). \tag{1}$$

The optimal parameters $\boldsymbol{\theta}^*$ are commonly obtained by minimizing the error of the forward or inverse dynamics model,

$$\boldsymbol{\theta}^*_{\text{for}} = \arg\min_{\boldsymbol{\theta}} \textstyle\sum_{i=0}^{N} \|\ddot{\boldsymbol{x}}_i - \hat{\boldsymbol{f}}\left(\boldsymbol{x}_i, \dot{\boldsymbol{x}}_i, \boldsymbol{u}_i; \boldsymbol{\theta}\right)\|^2 \tag{2}$$

$$\boldsymbol{\theta}^*_{\text{inv}} = \arg\min_{\boldsymbol{\theta}} \textstyle\sum_{i=0}^{N} \|\boldsymbol{u}_i - \hat{\boldsymbol{f}}^{\text{-1}}\left(\boldsymbol{x}_i, \dot{\boldsymbol{x}}_i, \ddot{\boldsymbol{x}}_i; \boldsymbol{\theta}\right)\|^2. \tag{3}$$

Depending on the chosen representation for $\boldsymbol{f}$, the model hypotheses spaces and the optimization method changes.

White-box Models These models use the analytical equations of motions to formalize the hypotheses space of $\boldsymbol{f}$ and the interpretable physical parameters such as mass, inertia or length as parameters $\boldsymbol{\theta}$. Therefore, white-box models are limited to describe the phenomena incorporated within the equations of motions but generalize to unseen state regions as the parameters are global. This approach was initially proposed for rigid-body chain manipulators by Atkeson et. al. [12]. Using the recursive Newton-Euler algorithm (RNEA) [17], the authors derived features that simplify the inference of $\boldsymbol{\theta}$ to linear regression. The resulting parameters must not be necessarily be physically plausible as constraints between the parameters exist. For example, the inertia matrix contained in $\boldsymbol{\theta}^*$ must be positive definite matrix and fulfill the triangle inequality. Since then, various parameterizations for the physical parameters have been proposed to enforce these constraints through the virtual parameters. Various reparameterizations [14], [18], [19] were proposed to guarantee physically plausible inertia matrices. Using these virtual parameters, the optimization does not simplify to linear regression but can be solved by unconstrained gradient-based optimization and is guaranteed to preserve physically plausibility.

Black-box Models These models are generic function approximators such as locally linear models [20], [21], Gaussian processes [22]–[24], deep- [25], [26] or graph networks [27] for $\boldsymbol{f}$. These approximators can fit arbitrary and complex dynamics with high fidelity but have an undefined behavior outside the training distribution and might be physically unplausible even on the training domain. Due to the local nature of the representation, the behavior is only well defined on the training domain and hence the learned models do not extrapolate well. Furthermore, these models can learn implausible system violating fundamental physics laws such as energy conservation. Only recently deep networks were augmented with knowledge from physics to constrain network representations to be physically plausible on the training domain [1]–[7]. However, the behavior outside the training domains remains unknown.

III. Models for Model-Based RL

For MBRL, black-box models have been widely adopted due to their generic applicability and simplicity [28]–[30]. In the following, we will elaborate on specific aspects of MBRL which make model learning for MBRL challenging, and questions the use of black-box over white-box structures.

Data Distribution MBRL is commonly applied to complex tasks which involve contacts of multiple bodies, such as object manipulation and locomotion. In this case, the training data lies on a complex manifold separating physically feasible and impossible states, e.g., object contact vs. penetration. In addition, the data is not uniformly distributed over the set of feasible states, but accumulated at the manifold boundaries. In the considered BiC task, the ball is mostly observed at a certain distance from the cup due to the string constraint, rarely closer and never further. This complex data manifold is in contrast to model learning for simper tasks where the data is evenly distributed in the feasible state region, which is the convex set of the training data.

Model Usage MBRL uses the model to plan trajectories and evaluate the policy. During the planning, the predicted trajectories can venture to physically impossible states and exploit potential shortcuts to improve control. This behavior is especially likely in constrained tasks where one needs to traverse along the edge of the feasible states. For example, to solve the BiC task, one needs to plan with the string maximally extended. In this configuration, the planned trajectory can easily diverge to states where the string-length would be longer than physically possible. Conversely, for model-based policies such impossible regions are no concern for the model. In this setting the model is not queried in these configurations as the system cannot enter these states without system damage, malfunction or erroneous measurements.

These two characteristics of MBRL affect the model representations differently. Black-box models are less adapt at learning models from highly localized data as they can only extrapolate locally. In particular, this local interpolation can fail at the boundaries where bodies are in contact. Ill-fitted boundaries make it very likely that the planned trajectories diverge to physically implausible regions and that the policy optimization exploits any shortcuts within these regions. More data cannot resolve the problem, as the data from the physically implausible regions cannot be obtained from the real-world system. In contrast, white-box models are less susceptible to the irregular data distribution due to the global structure. Furthermore, many strategies have been developed for white-box models to avoid physically implausible regions within the simulation community. For example, white-box models avoid implausible states by generating forces orthogonal to the violated constraint to push the system state back to the physically feasible states. Due to these advantages of white-box models, this model representation can be beneficial for MBRL applications and the underfitting, which limits the application to model-based policies, is only of secondary concern. To test this hypothesis, we consider the BiC task, which relies heavily on the string constraint. In the following we construct a generic differentiable white-box structure for such a nonholonomic constraint expressed in maximal coordinates.

IV. Differentiable Simulation Models

In the following two sections we describe the used differentiable simulator based on the Newton-Euler algorithm in terms of the elegant Lie algebra formulation [31]. First we describe the simulator for systems with holonomic constraints, i.e., kinematic chains, and then extend it to systems with nonholonomic constraints. In the following we will refer to these models as *DiffNEA* as these models are based on the differentiability of the Newton-Euler equation.

Rigid-Body Physics with Holonomic Constraints For rigid-body systems with holonomic constraints the system dynamics can expressed analytically in maximal coordinates $\boldsymbol{x}$, i.e., task space, and reduced coordinates $\boldsymbol{q}$, i.e., joint space. If expressed using maximal coordinates, the dynamics is a constrained problem with the holonomic constraints $g(\cdot)$. For the reduced coordinates, the dynamics are reparametrized such that the constraints are always fulfilled and the dynamics are unconstrained. Mathematically this is described by

$$\ddot{\boldsymbol{x}} = f(\boldsymbol{x}, \dot{\boldsymbol{x}}, \boldsymbol{u}; \boldsymbol{\theta}) \quad \text{s.t.} \quad g(\boldsymbol{x}; \boldsymbol{\theta}) = 0 \tag{4}$$

$$\Rightarrow \ddot{\boldsymbol{q}} = f(\boldsymbol{q}, \dot{\boldsymbol{q}}, \boldsymbol{u}; \boldsymbol{\theta}). \tag{5}$$

For model learning of such systems one commonly exploits the reduced coordinate formulation and minimizes the squared loss of the forward or inverse model. For kinematic trees the forward dynamics $\boldsymbol{f}(\cdot)$ can be easily computed using the articulated body algorithm (ABA) and the inverse dynamics $\boldsymbol{f}^{-1}(\cdot)$ via the recursive Newton-Euler algorithm (RNEA) [17]. Both algorithms are inherently differentiable and one can solve the optimization problem of Equation 2 using backpropagation. In this implementation, we use the Lie formulations of ABA and RNEA [31] for compact and intuitive compared to the initial derivations by [12], [17]. ABA and RNEA propagate velocities and accelerations from the kinematic root to the leaves and the forces and impulses from the leaves to the root. This propagation along the chain can be easily expressed in Lie algebra by

$$\bar{\boldsymbol{v}}_j = \mathrm{Ad}_{\boldsymbol{T}_{j,i}} \bar{\boldsymbol{v}}_i, \qquad \bar{\boldsymbol{a}}_j = \mathrm{Ad}_{\boldsymbol{T}_{j,i}} \bar{\boldsymbol{a}}_i, \tag{6}$$

$$\bar{\boldsymbol{l}}_j = \mathrm{Ad}^T_{\boldsymbol{T}_{j,i}} \bar{\boldsymbol{l}}_i, \qquad \bar{\boldsymbol{f}}_j = \mathrm{Ad}^T_{\boldsymbol{T}_{j,i}} \bar{\boldsymbol{f}}_i. \tag{7}$$

with the generalized velocities $\bar{\boldsymbol{v}}$, accelerations $\bar{\boldsymbol{a}}$, forces $\bar{\boldsymbol{f}}$, momentum $\bar{\boldsymbol{l}}$ and the adjoint transform $\mathrm{Ad}_{\boldsymbol{T}_{j,i}}$ from the ith to the jth link. The generalized entities noted by $\bar{\cdot}$ combine the linear and rotational components, e.g., $\bar{\boldsymbol{v}} = [\boldsymbol{v}, \boldsymbol{\omega}]$ with the linear velocity $\boldsymbol{v}$ and the rotational velocity $\boldsymbol{\omega}$. The Newton-Euler equation is described by

$$\bar{\boldsymbol{f}}_{\text{net}} = \bar{\boldsymbol{M}} \bar{\boldsymbol{a}} - \mathrm{ad}^*_{\bar{\boldsymbol{v}}} \bar{\boldsymbol{M}} \bar{\boldsymbol{v}},$$

$$\mathrm{ad}^*_{\bar{\boldsymbol{v}}} = \begin{bmatrix} [\boldsymbol{\omega}] & \boldsymbol{0} \\ [\boldsymbol{v}] & [\boldsymbol{\omega}] \end{bmatrix}, \quad \bar{\boldsymbol{M}} = \begin{bmatrix} \boldsymbol{J} & m[\boldsymbol{p}_m] \\ m[\boldsymbol{p}_m]^T & m\boldsymbol{I} \end{bmatrix}$$

with the inertia matrix $\boldsymbol{J}$, the link mass m, the center of mass offset $\boldsymbol{p}_m$. Combining this message passing with the Newton Euler equation enables a compact formulation of RNEA and ABA. The gradient based optimization also enables the reparametrization of the physical parameters with virtual parameters $\boldsymbol{\theta}_{\text{v}}$ that guarantee to be physically plausible [14], [18], [19].

Rigid-Body Physics with Nonholonomic Constraints For a mechanical system with nonholonomic constraints, the system dynamics cannot be expressed via an unconstrained equations with reduced coordinates. For the system

$$\ddot{\boldsymbol{x}} = f(\boldsymbol{x}, \dot{\boldsymbol{x}}, \boldsymbol{u}; \boldsymbol{\theta}) \quad \text{s.t.} \quad h(\boldsymbol{x}; \boldsymbol{\theta}) \leq 0, \quad g(\boldsymbol{x}, \dot{\boldsymbol{x}}; \boldsymbol{\theta}) = 0,$$

the constraints are nonholonomic as $h(\cdot)$ is an inequality constraint and $g(\cdot)$ depends on the velocity. Inextensible strings are an example for systems with inequality constraint, while the bicycle is a system with velocity dependent constraints. For such systems, one cannot optimize the unconstrained problem directly, but must identify parameters that explain the data and adhere to the constraints.

The dynamics of the constrained rigid body system can be described by the Newton-Euler equation,

$$\bar{\boldsymbol{f}}_{\text{net}} = \bar{\boldsymbol{f}}_g + \bar{\boldsymbol{f}}_c + \bar{\boldsymbol{f}}_{\boldsymbol{u}} = \bar{\boldsymbol{M}} \bar{\boldsymbol{a}} - \mathrm{ad}^*_{\bar{\boldsymbol{v}}} \bar{\boldsymbol{M}} \bar{\boldsymbol{v}}, \tag{8}$$

$$\Rightarrow \bar{\boldsymbol{a}} = \bar{\boldsymbol{M}}^{-1} \left(\bar{\boldsymbol{f}}_g + \bar{\boldsymbol{f}}_c + \bar{\boldsymbol{f}}_{\boldsymbol{u}} + \mathrm{ad}^*_{\bar{\boldsymbol{v}}} \bar{\boldsymbol{M}} \bar{\boldsymbol{v}} \right), \tag{9}$$

where the net force $\bar{\boldsymbol{f}}_{\text{net}}$ contains the gravitational force $\bar{\boldsymbol{f}}_g$, the constraint force $\bar{\boldsymbol{f}}_c$ and the control force $\bar{\boldsymbol{f}}_{\boldsymbol{u}}$. If one can differentiate the constraint solver computing the constraint force w.r.t. to the parameters, one can identify the parameters $\boldsymbol{\theta}$ via gradient descent. This optimization problem can be described by

$$\begin{aligned} \boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \textstyle\sum_{i=0}^{N} \| \bar{\boldsymbol{a}}_i - \bar{\boldsymbol{M}}^{-1}_{\boldsymbol{\theta}} \big(\bar{\boldsymbol{f}}_g(\boldsymbol{\theta}) \\ + \bar{\boldsymbol{f}}_c(\bar{\boldsymbol{x}}_i, \bar{\boldsymbol{v}}_i; \boldsymbol{\theta}) + \bar{\boldsymbol{f}}_{\boldsymbol{u}} + \mathrm{ad}^*_{\bar{\boldsymbol{v}}_i} \bar{\boldsymbol{M}}(\boldsymbol{\theta}) \bar{\boldsymbol{v}}_i \big) \|^2. \end{aligned} \tag{10}$$

For the inequality constraint, one can to reframe it as an easier equality constraint, by passing the function through a ReLU nonlinearity $\sigma(\cdot)$, so $g(\boldsymbol{x};\boldsymbol{\theta}) = \sigma(h(\boldsymbol{x};\boldsymbol{\theta})) = 0$. From a practical perspective, the softplus nonlinearity provides a soft relaxation of the nonlinearity for smoother optimization. Since this equality constraint should always be enforced, we can utilize our dynamics to ensure this on the derivative level, so $g(\cdot) = \dot{g}(\cdot) = \ddot{g}(\cdot) = 0$ for the whole trajectory. With this augmentation, the constraint may now be expressed as $\boldsymbol{g}(\boldsymbol{x}, \dot{\boldsymbol{x}}; \boldsymbol{\theta}) = \boldsymbol{0}$. The complete loss is described by

$$\begin{aligned}\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \textstyle\sum_{i=0}^{N} \|\bar{\boldsymbol{a}}_i - \boldsymbol{f}(\bar{\boldsymbol{x}}_i, \bar{\boldsymbol{v}}_i, \bar{\boldsymbol{u}}_i; \boldsymbol{\theta})\|^2 \\ + \lambda_g \|g(\boldsymbol{\theta})\|^2 + \lambda_{\dot{g}} \|\dot{g}(\boldsymbol{\theta})\|^2 + \lambda_{\ddot{g}} \|\ddot{g}(\boldsymbol{\theta})\|^2\end{aligned} \quad (11)$$

with the scalar penalty parameters λ_g, $\lambda_{\dot{g}}$ and $\lambda_{\ddot{g}}$.

V. Related Work

Differentiable simulators have been previously proposed for model-based reinforcement learning [32], [33] and planning [34]. In these works, the authors focus on the differentiability w.r.t. to the previous state and use the differentiable model to backpropagate in time to optimize policies or plans. Instead, we focus on the differentiability w.r.t. to model parameter and deploy the differentiable model for system identification of robotic system described using reduced and maximal coordinates as well as explicit holonomic and nonholonomic constraints. Such systems are the main interest of MBRL as the common task usually cannot be described using solely unconstrained reduced coordinates.

To obtain differentiable simulators, the main problem is differentiating through the constraint force solver computing $\boldsymbol{f}_c$. Various approaches have been proposed, e.g., Belbute-Peres et. al. [33] describe a method to differentiate through the common LCP solver of simulators, Geilinger et. al. [35] describe a smoothed frictional contact model and Hu et al. [36] describe a continuous collision resolution approach to improve the gradient computation. In this work we follow the approach of [32], [37] and use automatic differentiation to differentiate through the closed form solution of $\boldsymbol{f}_c$. For our considered task this closed form solution is possible and we do not need to rely on more complex approaches presented in literature.

VI. Experimental Setup

To evaluate the performance of white-box and black-box models for MBRL, we apply these model representations within an offline MBRL algorithm on the physical system to solve BiC. We test the models within an offline RL algorithm as this approach amplifies the challenges of model learning. In this setting, additional real-world data cannot be used to compensate for modeling errors. BiC is a common benchmark for real-world reinforcement learning and has been used multiple times for model-free reinforcement learning [8]–[10] as well as model-free iterative learning control [38]. Until now this task has *not* been solved on a physical system with MBRL as learning a reliable string model is challenging.

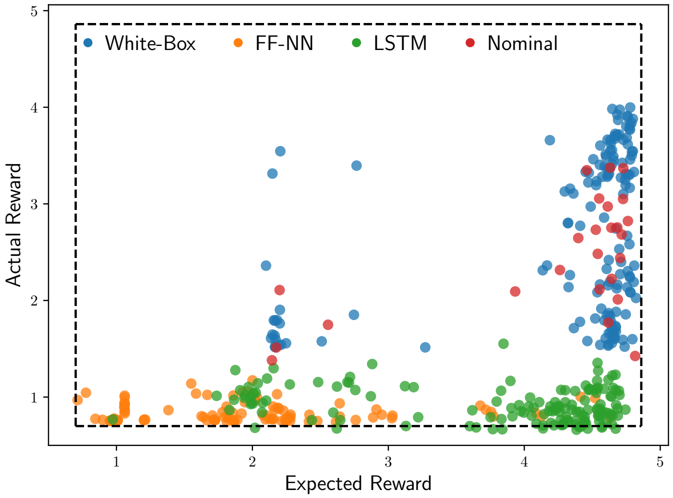

Fig. 2. Comparison of the expected reward and the actual reward on the MuJoCo simulator for the LSTM, the feed-forward neural network (FF-NN) as well as the nominal and learnt white-box model. The learnt and nominal white-box model achieve a comparable performance and solve the BiC swing-up for multiple seeds. Neither the LSTM nor the FF-NN achieve a single successful swing-up despite being repeated with 50 different seeds and using all the data of generated by the white-box models.

BiC Black-box Model A feedforward network (FF-NN) and a long short-term memory network (LSTM) [39] is used as black-box model. The networks model only the string dynamics and receive the task space movement of the last joint and the ball movement as input and predict the ball acceleration, i.e., $\ddot{\boldsymbol{x}}_B = f(\boldsymbol{x}_{J_4}, \dot{\boldsymbol{x}}_{J_4}, \ddot{\boldsymbol{x}}_{J_4}, \boldsymbol{x}_B, \dot{\boldsymbol{x}}_B)$.

BiC White-box Model For this model, the robot manipulator is modeled as a rigid-body chain using reduced coordinates. The ball is modeled via a constrained particle simulation with an inequality constraint. Both models are interfaced via the task space movement of the robot after the last joint. The manipulator model predicts the task-space movement after the last joint. The string model transforms this movement to the end-effector frame via $\boldsymbol{T}_E$ (Figure 1 a), computes the constraint force $\boldsymbol{f}_c$ and the ball acceleration $\ddot{\boldsymbol{x}}_B$. Mathematically this model is described by

$$\ddot{\boldsymbol{x}}_B = \tfrac{1}{m_B}\left(\boldsymbol{f}_g + \boldsymbol{f}_c\right), \quad (12)$$

$$g(\boldsymbol{x}; \boldsymbol{\theta}_S) = \sigma(\|\boldsymbol{x}_B - \boldsymbol{T}_E\, \boldsymbol{x}_{J_4}\|_2^2 - r^2) = 0, \quad (13)$$

where $\boldsymbol{x}_B$ is the ball position, $\boldsymbol{x}_{J_4}$ the position of the last joint and r the string length. In the following we will abbreviate $\boldsymbol{x}_B - \boldsymbol{T}_E\, \boldsymbol{x}_{J_4} = \Delta$ and the cup position by $\boldsymbol{T}_E\, \boldsymbol{x}_{J_4} = \boldsymbol{x}_C$. The constraint force can be computed analytically with the principle of virtual work and is described by

$$\boldsymbol{f}_c(\boldsymbol{\theta}_S) = -m_B\, \sigma'(z)\, \frac{\Delta^\intercal \boldsymbol{g} - \Delta^\intercal \ddot{\mathbf{x}}_C + \dot{\Delta}^\intercal \dot{\Delta}}{\Delta^\intercal \Delta + \delta} \quad (14)$$

with $z = \|\Delta\|_2 - r$, and the gravitational vector $\boldsymbol{g}$. When simulating the system, we set $\ddot{g} = -\boldsymbol{K}_p g - \boldsymbol{K}_d \dot{g} \leq 0$ to avoid constraint violations and add friction to the ball for numerical stability. This closed form constraint force is differentiable and hence one does not need to incorporate any special differentiable simulation variants.

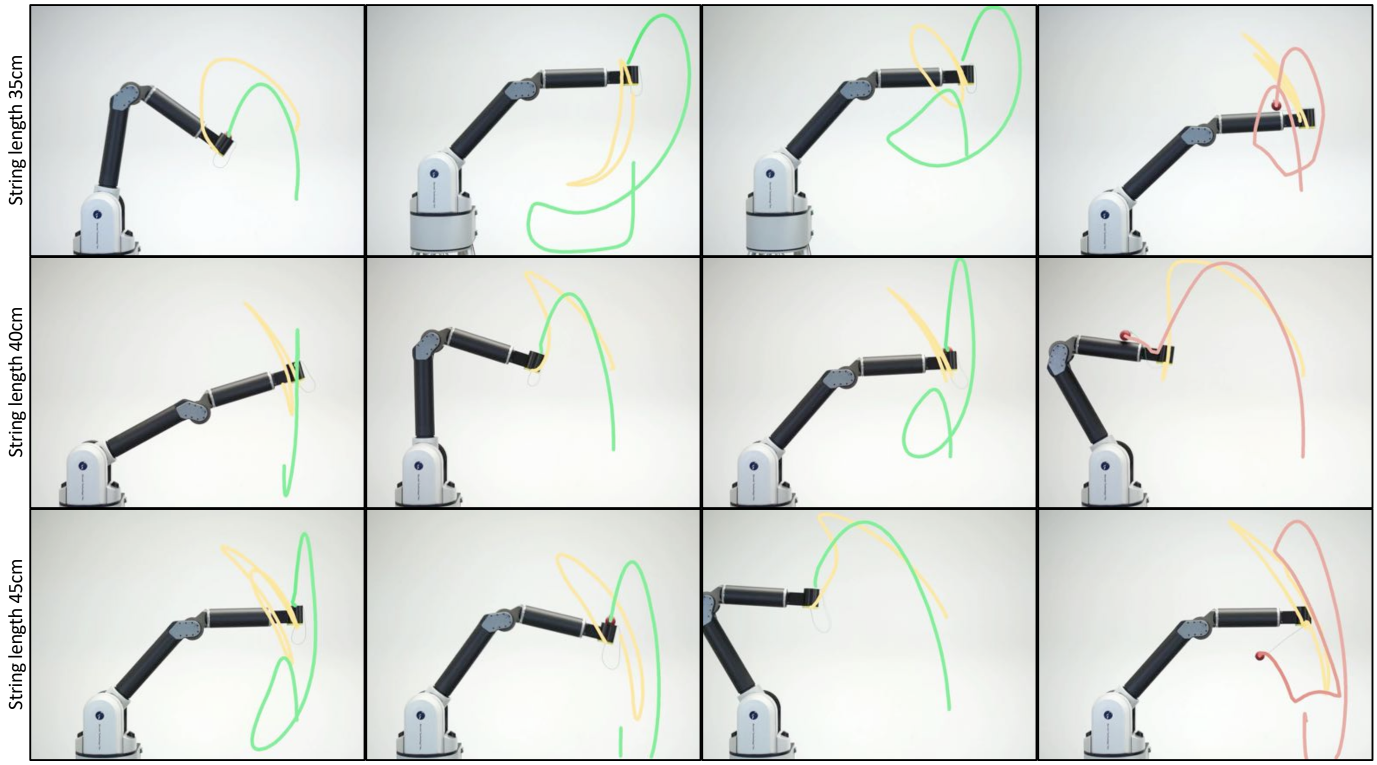

Fig. 3. Three different successful swing-ups for the three different string lengths using the DiffNEA White-Box model with eREPS for offline model-based reinforcement learning. This approach can learn different swing-ups from just 4 minutes of data, while all tested black-box models fail at the task. The different solutions are learned using different seeds. The unsuccessful trials of the DiffNEA model nearly solve the BiC tasks but the ball bounces off the cup or arm. Videos and pictures for all models and all experiments can be found at **https://sites.google.com/view/ball-in-a-cup-in-4-minutes/**

Offline Reinforcement Learning This RL problem formulation studies the problem of learning an optimal policy from a fixed dataset of arbitrary experience [40], [41]. Hence, the agent is bound to a dataset and cannot explore the environment. For solving this problem, we use a model-based approach were one first learns a model from the data and then performs episodic model-free reinforcement learning (MFRL) using this approximate model. For the model-free RL we use episodic relative entropy policy search (eREPS) with an additional KL-divergence constraint on the maximum likelihood policy update [42] and parameter exploration [43]. The policy is a probabilistic movement primitive (ProMP) [44], [45] describing a distribution over trajectories.

Dataset For the manipulator identification the robot executes a 40s high-acceleration sinusoidal joint trajectory (Figure 1 b). For the string model identification, the robot executes a 40s slow cosine joint trajectories to induce ball oscillation without contact with the manipulator (Figure 1 c). The ball trajectories are averaged over five trajectories to reduce the variance of the measurement noise. The training data does not contain swing-up motions and, hence the model must extrapolate to achieve the accurate simulation of the swing-up. The total dataset used for offline RL contains only 4 minutes of data. To simplify the task for the deep networks, the training data consists of the original training data plus all data generated by the white-box model during evaluation. Therefore, the network training data contains successful BiC tasks.

Reward The dense episodic reward is inspired by the potential of an electric dipole and augmented with regularizing penalties for joint positions and velocities. The complete reward is defined as

$$R(\boldsymbol{s}_{<N}) = \exp\left(\frac{1}{2}\max_t \psi_t + \frac{1}{2}\psi_N\right) \\ -\frac{1}{N}\sum_{i=0}^{N} \lambda_{\boldsymbol{q}} \|\boldsymbol{q}_i - \boldsymbol{q}_0\|_2^2 + \lambda_{\dot{\boldsymbol{q}}} \|\dot{\boldsymbol{q}}_i\|_2^2,$$

with $\psi_t = \Delta_t^{\intercal} \hat{\boldsymbol{m}}(\boldsymbol{q}_t) / \left(\Delta_t^{\intercal}\Delta_t + \epsilon\right)$ and the normal vector of the end-effector frame $\hat{\boldsymbol{m}}$ which depends on joint configuration $\boldsymbol{q}_t$. For the white-box model, the *predicted* end-effector frame is used during policy optimization. Therefore, the policy is optimized using the reward computed in the approximated model. The black-box models uses the true reward, rather than the reward bootstrapped from the learned model.

VII. Experimental Results

Videos documenting all experiments can be found at **https://sites.google.com/view/ball-in-a-cup-in-4-minutes/**.

Simulation Results The simulation experiments test the models with idealized observations from MuJoCo [46] and enable a quantitative comparison across many seeds. For each model representation, 15 different learned models are evaluated with 150 seeds for the MFRL. The average statistics of the best

TABLE I
OFFLINE REINFORCEMENT LEARNING RESULTS FOR THE BALL IN A CUP TASK, ACROSS BOTH SIMULATION AND THE PHYSICAL SYSTEM. LENGTH REFERS TO THE STRING LENGTH IN CENTIMETERS. REPEATABILITY IS REPORTED FOR THE BEST PERFORMING REINFORCEMENT LEARNING SEED.

	SIMULATION				PHYSICAL SYSTEM			
MODEL	LENGTH	AVG. REWARD	TRANSFERABILITY	REPEATABILITY	LENGTH	AVG. REWARD	TRANSFERABILITY	REPEATABILITY
LSTM	40CM	0.92 ± 0.37	0%	-	40CM	0.91 ± 0.56	0%	0%
FF-NN	40CM	0.86 ± 0.35	0%	-	40CM	1.46 ± 0.78	0%	0%
NOMINAL	40CM	2.45 ± 1.15	**64%**	-	40CM	1.41 ± 0.45	0%	0%
DIFFNEA	40CM	$\mathbf{2.73 \pm 1.64}$	52%	-	40CM	$\mathbf{1.77 \pm 0.74}$	**60%**	90%
					35CM	1.58 ± 0.15	30%	70%
					45CM	1.74 ± 0.71	**60%**	**100%**

ten reinforcement learning seeds are shown in Table I and the expected versus obtained reward is shown in Figure 2 (a).

The DiffNEA white-box model is able to learn the BiC swing-up for every tested model. The transferability to the MuJoCo simulator depends on the specific seed, as the problem contains many different local solutions and only some solutions are robust to slight model variations. The MuJoCo simulator is different from the DiffNEA model as MuJoCo simulates the string as a chain of multiple small rigid bodies. The performance of the learned DiffNEA is comparable to the performance of the DiffNEA model with the nominal values.

The FF-NN and LSTM black-box models do not learn a single successful transfers despite being tried on ten different models and 150 different seeds, using additional data that includes swing-ups and observing the real instead of the imagined reward. These learned models cannot stabilize the ball beneath the cup. The ball immediately diverges to a physical unfeasible region. The attached videos compare the real (red) vs. imagined (yellow) ball trajectories. Within the impossible region the policy optimizer exploits the random dynamics where the ball teleports into the cup. Therefore, the policy optimizers converges to random movements.

Real-Robot Results The experiments are performed using the Barrett WAM and three different string-lengths, i.e., 35cm, 40cm and 45cm. For each model a 50 different seeds are evaluated on the physical system. A selection of the of trials using the learned DiffNEA white-box model is shown in Figure 3. The average statistics of the best ten seeds are summarized in Table I.

The DiffNEA white-box model is capable of solving BiC using offline MBRL for all string-lengths. This approach obtains very different solutions that transfer to the physical system. Some solutions contain multiple pre-swings which show the quality of the model for long-planning horizons. The best movements also repeatedly achieve the successful task completion. Solutions that do not transfer to the system, perform feasible movements where the ball bounces of the cup rim. The nominal DiffNEA model with the measured arm and string parameters does not achieve a successful swing-up. The ball always overshoots and bounces of the robot-arm for this model. None of the tested black-box models achieve the BiC swing-up despite using more data and the true rewards during planning. Especially the FF-NN model converges to random policies, which result in ball movement that do not even closely resemble a potential swing-up. The convergence to very different movements shows that the models contain multiple shortcuts capable of teleporting the imagined ball into the cup.

VIII. CONCLUSION & FUTURE WORK

In this paper we argue that for highly constrained tasks, white-box models provide a benefit over black-box model for MBRL, and verify this hypothesis through an extensive evaluation on ball in a cup task on a real robotic platform. The ball in a cup task shows that guaranteed physically plausible models are preferable compared to deep networks for this task. The white-box DiffNEA model solves BiC with only four minutes of data via offline MBRL. All network models fail on this task. For MBRL the inherent underfitting of white-box models for real-world systems might only be of secondary concern compared to the detrimental effect of divergence to physically unfeasible states. In addition, we extend the existing methods for identification of physics parameters to systems with maximal coordinates and nonholonomic inequality constraints. The real-world experiments show that this approach is also applicable for real-world systems that include unmodeled physical phenomena, such as cable drives and stiction. In future work, we want to look at grey-box models as well as robust policy optimization.

Grey-box Models This model representation combines black-box and white-box models to achieve high-fidelity approximations of complex physical phenomena with guaranteed avoidance of impossible state regions. Currently, various initial variants [13], [24], [47], [48] exist, but a principled method that optimizes the black- and white-box parameters simultaneously remains an open question.

Robust Policy Optimization To improve the transferability of the learned optimal policies, robustness w.r.t. to model uncertainty needs to be incorporated into the policy optimization. Within this work we did not incorporate robustness in the policy optimization but plan to extend the DiffNEA model to probabilistic DiffNEA models with domain randomization [49]–[51], which is only applicable to white-box models.

References

[1] M. Lutter, C. Ritter, and J. Peters, "Deep Lagrangian Networks: Using physics as model prior for deep learning," in *International Conference on Learning Representations (ICLR)*, 2019.

[2] M. Lutter and J. Peters, "Deep Lagrangian Networks for end-to-end learning of energy-based control for under-actuated systems," in *International Conference on Intelligent Robots and Systems (IROS)*, 2019.

[3] S. Greydanus, M. Dzamba, and J. Yosinski, "Hamiltonian neural networks," in *Advances in Neural Information Processing Systems (NeuRIPS)*, 2019.

[4] J. K. Gupta, K. Menda, Z. Manchester, and M. J. Kochenderfer, "A general framework for structured learning of mechanical systems," *arXiv preprint arXiv:1902.08705*, 2019.

[5] M. Cranmer, S. Greydanus, S. Hoyer, P. Battaglia, D. Spergel, and S. Ho, "Lagrangian neural networks," *arXiv preprint arXiv:2003.04630*, 2020.

[6] S. Saemundsson, A. Terenin, K. Hofmann, and M. Deisenroth, "Variational integrator networks for physically structured embeddings," in *International Conference on Artificial Intelligence and Statistics (Aistats)*, 2020.

[7] Y. D. Zhong, B. Dey, and A. Chakraborty, "Symplectic ode-net: Learning hamiltonian dynamics with control," *arXiv preprint arXiv:1909.12077*, 2019.

[8] J. Kober and J. R. Peters, "Policy search for motor primitives in robotics," in *Advances in Neural Information Processing Systems (NeuRIPS)*, 2009.

[9] D. Schwab, T. Springenberg, M. F. Martins, T. Lampe, M. Neunert, A. Abdolmaleki, T. Herkweck, R. Hafner, F. Nori, and M. Riedmiller, "Simultaneously learning vision and feature-based control policies for real-world ball-in-a-cup," *arXiv preprint arXiv:1902.04706*, 2019.

[10] P. Klink, H. Abdulsamad, B. Belousov, and J. Peters, "Self-paced contextual reinforcement learning," *Conference on Robot Learning (CoRL)*, 2019.

[11] A. M. Bloch, *Nonholonomic mechanics and control*. Springer, 2003.

[12] C. G. Atkeson, C. H. An, and J. M. Hollerbach, "Estimation of inertial parameters of manipulator loads and links," *The International Journal of Robotics Research (IJRR)*, 1986.

[13] M. Lutter, J. Silberbauer, J. Watson, and J. Peters, "A differentiable newton euler algorithm for multi-body model learning," in *Robotics: Science and Systems Conference (RSS), Workshop on Structured Approaches to Robot Learning for Improved Generalization*, 2020.

[14] G. Sutanto, A. Wang, Y. Lin, M. Mukadam, G. Sukhatme, A. Rai, and F. Meier, "Encoding physical constraints in differentiable Newton-Euler Algorithm," *Learning for Dynamics Control (L4DC)*, 2020.

[15] L. B. Rall, *Automatic Differentiation: Techniques and Applications*, ser. Lecture Notes in Computer Science. Springer, 1981, vol. 120.

[16] K. J. Åström and P. Eykhoff, "System identification—a survey," *Automatica*, 1971.

[17] R. Featherstone, *Rigid Body Dynamics Algorithms*. Springer-Verlag, 2007.

[18] S. Traversaro, S. Brossette, A. Escande, and F. Nori, "Identification of fully physical consistent inertial parameters using optimization on manifolds," in *International Conference on Intelligent Robots and Systems (IROS)*, 2016.

[19] P. M. Wensing, S. Kim, and J.-J. E. Slotine, "Linear matrix inequalities for physically consistent inertial parameter identification: A statistical perspective on the mass distribution," *IEEE Robotics and Automation Letters (RAL)*, 2017.

[20] C. G. Atkeson, A. W. Moore, and S. Schaal, "Locally weighted learning for control," *Artificial Intelligence Review*, 1997.

[21] S. Schaal, C. G. Atkeson, and S. Vijayakumar, "Scalable techniques from nonparametric statistics for real time robot learning," *Applied Intelligence*, 2002.

[22] J. Kocijan, R. Murray-Smith, C. E. Rasmussen, and A. Girard, "Gaussian process model based predictive control," in *American Control Conference*, 2004.

[23] D. Nguyen-Tuong, M. Seeger, and J. Peters, "Model learning with local gaussian process regression," *Advanced Robotics*, 2009.

[24] D. Nguyen-Tuong and J. Peters, "Using model knowledge for learning inverse dynamics." in *International Conference on Robotics and Automation*, 2010.

[25] M. Jansen, "Learning an accurate neural model of the dynamics of a typical industrial robot," in *International Conference on Artificial Neural Networks*, 1994.

[26] A. Sanchez-Gonzalez, N. Heess, J. T. Springenberg, J. Merel, M. Riedmiller, R. Hadsell, and P. Battaglia, "Graph networks as learnable physics engines for inference and control," *arXiv preprint arXiv:1806.01242*, 2018.

[27] A. Sanchez-Gonzalez, J. Godwin, T. Pfaff, R. Ying, J. Leskovec, and P. W. Battaglia, "Learning to simulate complex physics with graph networks," *arXiv preprint arXiv:2002.09405*, 2020.

[28] M. Janner, J. Fu, M. Zhang, and S. Levine, "When to trust your model: Model-based policy optimization," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[29] K. Chua, R. Calandra, R. McAllister, and S. Levine, "Deep reinforcement learning in a handful of trials using probabilistic dynamics models," in *Advances in Neural Information Processing Systems (NeuRIPS)*, 2018.

[30] E. Langlois, S. Zhang, G. Zhang, P. Abbeel, and J. Ba, "Benchmarking model-based reinforcement learning," *arXiv preprint arXiv:1907.02057*, 2019.

[31] J. Kim, "Lie group formulation of articulated rigid body dynamics," Carnegie Mellon University, Tech. Rep., 2012.

[32] J. Degrave, M. Hermans, J. Dambre *et al.*, "A differentiable physics engine for deep learning in robotics," *Frontiers in neurorobotics*, vol. 13, p. 6, 2019.

[33] F. de Avila Belbute-Peres, K. Smith, K. Allen, J. Tenenbaum, and J. Z. Kolter, "End-to-end differentiable physics for learning and control," in *Advances in Neural Information Processing Systems*, 2018, pp. 7178–7189.

[34] M. A. Toussaint, K. R. Allen, K. A. Smith, and J. B. Tenenbaum, "Differentiable physics and stable modes for tool-use and manipulation planning," 2018.

[35] M. Geilinger, D. Hahn, J. Zehnder, M. Bächer, B. Thomaszewski, and S. Coros, "Add: Analytically differentiable dynamics for multi-body systems with frictional contact," *arXiv preprint arXiv:2007.00987*, 2020.

[36] Y. Hu, L. Anderson, T.-M. Li, Q. Sun, N. Carr, J. Ragan-Kelley, and F. Durand, "Difftaichi: Differentiable programming for physical simulation," *arXiv preprint arXiv:1910.00935*, 2019.

[37] E. Heiden, D. Millard, H. Zhang, and G. S. Sukhatme, "Interactive differentiable simulation," *arXiv preprint arXiv:1905.10706*, 2019.

[38] M. Bujarbaruah, T. Zheng, A. Shetty, M. Sehr, and F. Borrelli, "Learning to play cup-and-ball with noisy camera observations," *arXiv preprint arXiv:2007.09562*, 2020.

[39] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, 1997.

[40] S. Levine, A. Kumar, G. Tucker, and J. Fu, "Offline reinforcement learning: Tutorial, review, and perspectives on open problems," *arXiv preprint arXiv:2005.01643*, 2020.

[41] S. Lange, T. Gabel, and M. Riedmiller, "Batch reinforcement learning," in *Reinforcement Learning: State-of-the-Art*. Springer Berlin Heidelberg, 2012.

[42] K. Ploeger, M. Lutter, and J. Peters, "High acceleration reinforcement learning for real-world juggling with binary rewards," in *Conference on Robot Learning (CoRL)*, 2020.

[43] M. P. Deisenroth, G. Neumann, J. Peters *et al.*, "A survey on policy search for robotics," *Foundations and Trends® in Robotics*, 2013.

[44] A. Paraschos, C. Daniel, J. R. Peters, and G. Neumann, "Probabilistic movement primitives," in *Advances in neural information processing systems (NeurIPS*, 2013.

[45] A. Paraschos, C. Daniel, J. Peters, and G. Neumann, "Using probabilistic movement primitives in robotics," *Autonomous Robots*, 2018.

[46] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *International Conference on Intelligent Robots and Systems (IROS)*, 2012.

[47] J. Hwangbo, J. Lee, A. Dosovitskiy, D. Bellicoso, V. Tsounis, V. Koltun, and M. Hutter, "Learning agile and dynamic motor skills for legged robots," *Science Robotics*, vol. 4, no. 26, 2019.

[48] A. Allevato, E. S. Short, M. Pryor, and A. Thomaz, "Tunenet: One-shot residual tuning for system identification and sim-to-real robot task transfer," in *Conference on Robot Learning (CoRL)*, 2020.

[49] F. Ramos, R. C. Possas, and D. Fox, "Bayessim: adaptive domain randomization via probabilistic inference for robotics simulators," *arXiv preprint arXiv:1906.01728*, 2019.

[50] F. Muratore, M. Gienger, and J. Peters, "Assessing transferability from simulation to reality for reinforcement learning," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019.

[51] Y. Chebotar, A. Handa, V. Makoviychuk, M. Macklin, J. Issac, N. Ratliff, and D. Fox, "Closing the sim-to-real loop: Adapting simulation randomization with real world experience," in *International Conference on Robotics and Automation (ICRA)*, 2019.