Fundamental Challenges in Deep Learning for Stiff Contact Dynamics

Mihir Parmar*, Mathew Halm*†, and Michael Posa

Abstract—Frictional contact has been extensively studied as the core underlying behavior of legged locomotion and manipulation, and its discontinuous nature makes planning and control difficult even when an accurate model of the robot is available. Here, we make an empirical case that learning an accurate model in the first place can be confounded by contact-induced discontinuity, as modern deep learning approaches are inductively biased towards smooth motion. In a simulated experimental setting, we measure the effects of approximating discontinuous impact with varying degrees of stiffness. We find that even for a simple system, stiffness alone dramatically degrades long-term prediction and learned models’ fit to training data. Our results suggest that the data-efficiency of learning-based robotics and control methods are fundamentally held back by the dynamical properties of contact. Significant further investigation will be necessary to fully understand and mitigate these effects, and we suggest several avenues for future study.

I. INTRODUCTION

Accurate dynamics models, or transition models, which capture how robots move and interact with their environments, have been essential tools for enabling state-of-the-art accomplishments in robotic manipulation and locomotion. As the field moves from tightly-controlled laboratory experiments to unknown and unstructured environments, it is increasingly important for robots to identify appropriate models of their environment quickly. Despite recent successes leveraging modern deep learning to enable data-efficient learning and control (e.g. [1], [2], [3], [4], [5]), robotic performance remains decidedly sub-human in almost all scenarios. Theoretical results in physics-based dynamics ([6]) and machine learning ([7]) together suggest that the performance of many of these methods will suffer from fundamental contradictions between their inductive biases and the discontinuous nature of frictional contact. We make an empirical argument that this theoretical conflict has significant practical ramifications.

Classical system identification methods achieve data efficiency by learning a physics-based model, where the number of unknown parameters is far outnumbered by the quantity of data. In this underparameterized setting, the limited expressiveness of the model prevents overfitting to noisy data. However, these methods require detailed knowledge of the robot’s environment that is often unavailable (e.g. object properties in manipulation and terrain geometry in locomotion). Furthermore, they often embed restrictive, imperfect assumptions such as object rigidity and inelastic impact [8]. By contrast, some recent approaches fit a deep neural network (DNN) directly to the robot’s equations of motion (e.g. [2], [4], [9] and many others). A fundamental advantage of DNN’s over classical approaches is their ability to approximate any continuous function with arbitrary precision; thus they can be used readily in unknown and unstructured environments, and are not restricted by physics-based assumptions. A corresponding disadvantage, however, is that many distinct DNN parameterizations may exactly fit noisy data; to prevent overfitting, an appropriate inductive bias must be used to select parameters that reject noise and

*These authors contributed equally
† Corresponding author
All authors are with the GRASP Laboratory, University of Pennsylvania, Philadelphia, PA 19104, USA
{mhalm,mihir98,posa}@seas.upenn.edu
generalize to unseen states and inputs.

The essential inductive bias of deep learning is to select the smoothest interpolator of the data. This bias is not only an inherent predisposition of common training techniques like stochastic gradient descent (SGD) [7]; ubiquitous regularizers such as weight-decay and spectral normalization explicitly penalize or constrain Lipschitz constants of neural networks [10], [11]. Furthermore, enforced Lipschitz continuity of the equations of motion has been specifically studied for model-based reinforcement learning [12]. By contrast, physics-based models of robots are non-smooth, as discontinuity [6] or extreme stiffness [13] due to frictional impacts are fundamental features of many manipulation and legged locomotion tasks. The direct implication of this theoretical conflict is that much of deep learning machinery is biased against finding accurate models of robot motion. Another important downstream effect of discontinuity is numerical sensitivity in the training process, as small changes in parameters result in drastically different long-term predictions and losses [14], [15]. Because SGD’s behavior is inherently noisy, obtaining even an approximate interpolator of the training data may not be practically feasible in the face of poor conditioning. These ramifications can be observed even on 1D examples (Fig. 1), and recent results suggest that naïve DNN-based approaches struggle especially with long-term prediction [16], [17], [14]. Furthermore, in our previous work ([14]), we developed a structured approach for contact dynamics that circumvents the numerical challenges of discontinuity, which yielded a 100x increase in data efficiency compared to an unstructured DNN baseline. This work takes the important step of isolating the degree to which these dramatic performance gains can be attributed to the effects of discontinuity itself.

In this paper, we demonstrate that the sample complexity and training process can be directly degraded by the stiffness generated by frictional contact. In Section II we select a low-dimensional, simulated system, for which we can directly modify the degree to which discontinuity is approximately simulated by varying a stiffness setting. In Sections III and IV we describe a series of experiments to isolate and characterize the effect of stiffness on learned dynamics models’ predictive performance. We find that changes in stiffness alone can account for a 10x reduction in data efficiency and 20x increase in converged loss on the training set. Finally, in Section V we discuss the implications of our results, and briefly list related works and potential future directions.

II. EXAMPLE SYSTEM

Because deep learning is a fundamentally stochastic process with few theoretical guarantees, we choose to empirically assess the effects of stiffness on deep learning machinery with a controlled, tractable, and repeatable setup. This is in contrast to many experiments and benchmarking tasks, as they are often made to be relatively challenging through significant, unknown noise (inherent in all real world setups) or high dimensions (e.g. the humanoid tasks in the OpenAI gym [18]).

We therefore choose to conduct our experiments in simulation. While capturing and mitigating uncertainty is a key capability for robotics, analysis of deterministic simulation provides ideal experiment repeatability and an optimistic view of how robot learning algorithms may perform when sensing and process noise are well controlled. We select MuJoCo [13], because it allows for control over contact stiffness parameters, enabling comparison of the same system under different degrees of smoothness. While our ultimate goal is to understand how learning algorithms perform in the real world, we additionally value the relationship between MuJoCo’s particular approximation of real-world dynamics and learning performance, as the simulator is used in ubiquitous benchmarking suites [18], [19].

We follow previous studies [17], [16], [14], and choose a “die roll” system, in which a single, rigid cube makes contact with the ground. Like these works, we select the system for ease of computation; computational and sample complexity associated with high dimensions are avoided, as the cube’s configuration has only 6 degrees of freedom. However, while the dimension is small, the cube exhibits many of the hallmark challenges in contact modeling: stick-slip transition, discontinuous impact, multiple contact points, and extreme sensitivity to initial conditions; Figure 2 illustrates some of these behaviors in 2D.

Our 3D die system has a 13-dimensional state

where $p_t \in \mathbb{R}^3$ is the center of mass position; $q_t \in \mathbb{R}^4$ is the orientation of the cube, expressed as a quaternion; $p_t \in \mathbb{R}^3$ is the world-frame c.o.m. velocity; and $\omega_t \in \mathbb{R}^3$ is the body-frame angular velocity. Physical modeling predicts the discrete-time system dynamics $x_{t+1} = f(x_t)$ for a symmetric
here the stiffness $k$ is the primary mechanism resisting penetration, and the damping can be understood to control restitution. This system is intuitively similar to the mass-spring-damper system, though we note that the units of $k$ are $\frac{N}{m}$, whereas spring stiffness is typically expressed in $\frac{N}{m}$ units. MuJoCo’s default values for $k$ are in the 2000–2500 $\frac{N}{km}$ range. However, as we will discuss in Section III, the corresponding contact behavior is far softer than that of many real-world objects, including many robots.

### III. EXPERIMENT DESIGN

We now describe the process of learning a dynamical system from data; different challenges that stiffness can impose in a dynamics learning problem; and motivate the design choices for our experiments. The associated Pytorch codebase is available on GitHub[4].

#### A. Representing System Dynamics with Neural Networks

Many modern deep learning-based modeling approaches (e.g. [4], [3], [2], [15], [1]) follow the same fundamental approach: fitting a neural network directly to data collected from a dynamical system. In our case, we fit DNNs to simulated die roll trajectories. In line with MuJoCo’s physical modeling as outlined in Section II, we train neural networks to predict the next-state $x_{t+1}$, given a history of previous states $x_{t-h:t}$, where $h$ is the history length. We use the class of Recurrent Neural Networks (RNN) in our problem. While some implementations pick $h=0$ and map $x_t$ to $x_{t+1}$ with a simple multilayer perceptron ([2], [4]), using RNNs with $h>0$ is more flexible, and has been shown promise on modeling sequential trajectory data in robotics [15], [1]. Specifically, we train a RNN to predict the velocity of the die at next time-step $v_{t+1} = (p_{t+1}, \omega_{t+1})$ given a sequence of previous states $x_{t-h:t}$. Then we use the finite difference [4] to construct the full-state $x_{t+1} = [p_{t+1}; q_{t+1}; v_{t+1}]$. RNNs maintain a hidden state $z_t$ with initial value $0$. $z_t$ is updated sequentially for each $x_t$, as $z_{t+1} = \phi(x_t, z_t)$ from the previous hidden-state $z_{t-1}$, where $\phi$ is a learned non-linear function.

By recursively unfolding,

$$z_t = \phi(x_t, \phi(x_{t-1}, \phi(x_{t-2}, \ldots \phi(x_{t-h}, 0), \ldots)))$$  

Finally, we use a two-layer fully-connected network as a decoder $\phi_{dec}$ to extract the predicted velocity vector as $v_{t+1} = \phi_{dec}(z_t)$. We denote the entire network comprising of the RNN and the fully-connected layers as $v_{t+1} = f_d(x_{t-h:t})$, where $\theta$ represents the weight vector of the network.

#### B. Training Process

1) Data Generation: We generate a dataset $\{\tau\}$ of trajectories $\tau = \{x_0, x_1, x_2, \ldots, x_{T-1}\}$ of length $T$ by repeatedly simulating the die roll system in MuJoCo starting from random initial states until it comes to rest. Initial states are sampled uniformly around a nominal state $x_{0,ref}$, with position sampled from $p_{0,ref} \sim \mathcal{N}([-l, l]^3); q_{0,ref}$ rotated by an angle sampled from $\mathcal{U}(0, 2\pi)$ around an axis sampled from $\mathcal{U}([-1, 1]^3)$ followed by normalization; and velocity sampled from $v_{0,ref} \sim \mathcal{U}([-l, l]^3)$. We use three different settings of stiffness and damping parameters in MuJoCo, which control the contact behavior as discussed in [3] to capture different degrees of smoothness in our data. We vary the stiffness parameter (k) while holding the damping ratio of 0.1 constant by changing $b$. An exact listing of these parameter values is given in Table I. To evaluate physical realism of the data corresponding to each of these settings, we compute the maximum ground penetration of the die, averaged over

#### TABLE I: Die Roll System Parameters

<table>
<thead>
<tr>
<th>constant</th>
<th>symbol</th>
<th>value</th>
<th>units</th>
</tr>
</thead>
<tbody>
<tr>
<td>mass</td>
<td>$m$</td>
<td>0.37</td>
<td>kg</td>
</tr>
<tr>
<td>inertia</td>
<td>$I$</td>
<td>$6.167 \times 10^{-4}$</td>
<td>kgm$^2$</td>
</tr>
<tr>
<td>side length</td>
<td>$l$</td>
<td>0.1</td>
<td>m</td>
</tr>
<tr>
<td>gravity</td>
<td>$g$</td>
<td>9.81</td>
<td>m/s$^2$</td>
</tr>
<tr>
<td>friction coefficient</td>
<td>$\mu$</td>
<td>1</td>
<td>(unitless)</td>
</tr>
<tr>
<td>time-step</td>
<td>$\Delta t$</td>
<td>$6.74 \times 10^{-3}$</td>
<td>s</td>
</tr>
</tbody>
</table>

trajectories, from each setting. We refer to the three settings as Hard, Medium and Soft based on the stiffness setting and demonstrated ground penetration. Even for the Hard setting, which has a stiffness value comparable to MuJoCo’s default, we observe ground penetration of around 10% of the die body-length, larger than what we would expect in a real-world setting. Figure [3] provides a visualization for the amount of ground penetration observed corresponding to a trajectory from each of the three stiffness setting. For each of the three different stiffness settings, we collect 10,000 trajectories \{τ\}_\text{train} for hyperparameter optimization and training purposes, and 1,000 more trajectories \{τ\}_\text{eval} for evaluation of the optimized models.

2) Training a Model: To train one of our networks, we first aggregate a set of \(N\) trajectories \{\(\tau\)_\text{i}\} \(\) randomly sampled from \{\(\tau\)_\text{train}\} and slice them into training data inputs \(\{x_{t-h:t}\}\) and corresponding output labels \(\{v_{t+1}\}\). Since the average trajectory length for the Hard setting is of 80 time-steps, we truncate the Medium and Soft trajectories to 80 time-steps to make the amount of data per trajectory equal across all settings. To improve numerical conditioning during training, we follow a standard procedure of normalizing the input data to have zero mean and unit variance [4]. We further split the sliced data \(\{x_{t-h:t}, v_{t+1}\}\) in 70:20:10 proportions to \(D_{\text{train}}, D_{\text{val}}\) and \(D_{\text{test}}\).

We train the dynamics model \(f_{\theta}\) on \(D_{\text{train}}\) by minimizing the mean squared error (MSE)

\[
\mathcal{L}(\theta) = \frac{1}{|D|} \sum_{(x_{t-h:t}, v_{t+1}) \in D} ||v_{t+1} - f_{\theta}(x_{t-h:t})||^2, \tag{9}
\]

using the Adam optimizer [20]. We save the model with the lowest MSE over \(D_{\text{val}}\), and terminate training when it does not improve for 30 consecutive epochs. MSE over \(D_{\text{test}}\) is then used to as the metric during hyperparameter optimization.

3) Hyperparameter Optimization: Since data corresponding to different stiffness settings may cause the conditioning of the Adam optimization problem to be different, we perform brute-force search to tune the hyperparameter values separately for each stiffness setting. Precisely, for each stiffness setting, we consider three RNN variants for our network: Long Short Term Memory (LSTM) [21], Gated Recurrent Unit (GRU) [22] and Bi-directional Long Short Term Memory (BiLSTM) [23]. For each variant, we sweep over different values of learning-rate, hidden-layer size, history-length and weight-decay, centered around hand-tuned values. Table [III] provides the space of hyperparameters swept over in this process. For each combination of settings, we complete at least 10 training runs on high data regime of 500 example trajectories, and then select the setting with the lowest average MSE over \(D_{\text{val}}\). Table [IV] specifies the final set of selected hyperparameter values for each of the stiffness setting.

C. Measuring Stiffness’s Effect on Learning Performance

To perform an optimistic analysis on how well learning algorithms perform on systems with different stiffnesses, we analyze performance of only the hyperparameter-optimized models in three settings: data efficiency of long-term prediction; effectiveness of Adam in minimizing training-set loss; and effects of regularization on test-set loss.

Since deep learning is biased towards fitting a smooth interpolator on the data, we would expect that fitting a DNN on data collected from stiffer settings will suffer worse generalization capability compared to smoother systems when trained on the same quantity of data. We examine this hypothesis by comparing the prediction quality of models trained with different stiffness on different quantities of data. While we have followed a commonly used approach by training our models on single-step predictions (e.g. [3], [4]), the long-term prediction quality is essential for model-based control methods, such as MPC [3]. We therefore evaluate long-term prediction capability via temporally averaged absolute error over a model rollout as used in [14]. For a particular ground-truth trajectory \(\tau \in \{\tau\}_\text{eval}\), we use the initial \(h\) ground-truth states \(\{x_t\}_{t=h-1} \in \tau\) as input for the learned model and recursively construct predicted trajectory for next \(\hat{T}\) time-steps \(\{\hat{x}_t\}_{t=h+\hat{T}}\). We compute our metrics as

\[
e_{\text{pos}} = \frac{1}{h+\hat{T}} \sum_{j=h}^{h+\hat{T}} ||\hat{p}_j - p_j||_2, \tag{10}
\]

\[
e_{\text{rot}} = \frac{1}{h+\hat{T}} \sum_{j=h}^{h+\hat{T}} \text{angle}(\hat{q}_j, q_j), \tag{11}
\]

\[
e_{\text{vel}} = \frac{1}{h+\hat{T}} \sum_{j=h}^{h+\hat{T}} ||\hat{v}_j - v_j||^2, \tag{12}
\]

To make a fair evaluation for models with different history length and simultaneously ensure that the prediction horizon

\[
\begin{array}{c|c|c|c}
\text{Hyperparameter} & \text{Values} \\
\hline
\text{learning-rate} & 1e-3 & 1e-4 & 1e-5 \\
\text{hidden-size} & 128 & 256 & 512 \\
\text{history-length} & 4 & 8 & 16 \\
\text{weight-decay} & 0 & 4e-5 & 4e-3 \\
\end{array}
\]

\[
\begin{array}{c|c|c|c}
\text{Hyperparameter} & \text{Stiffness Setting} \\
\hline
\text{RNN Variant} & \text{Hard} & \text{Medium} & \text{Soft} \\
\hline
\text{learning-rate} & GRU & GRU & GRU \\
\text{hidden-size} & 128 & 128 & 128 \\
\text{history-length} & 16 & 16 & 16 \\
\text{weight-decay} & 0 & 4e-5 & 4e-5 \\
\end{array}
\]
is long enough to capture ground impact and block tumbling, we use $T = 50$ in our experiments.

Next, since naive prediction loss can be poorly conditioned when learning stiff contact-driven dynamics [6], we hypothesize that Adam will have difficulty in converging to good minima consistently; therefore, the training loss at convergence of stiffer models is likely to have higher mean and variation. We test this hypothesis by analyzing the converged training loss of multiple training runs of models across different stiffness and dataset sizes.

Finally, we consider regularization, specifically weight decay, which is intended to achieve better generalization to testing data via penalizing non-smoothness [10]. Therefore, we would expect that learned models are biased away from the true, stiff dynamics. We therefore hypothesize that for stiffer systems, regularization will have a greater negative effect on test set loss. We analyze this relationship by training our models across increasing regularization strength by increasing weight-decay coefficients and observe the change in test loss when compared to models trained with zero weight decay.

IV. RESULTS

We capture stiffness’s effects on long-term prediction in Figure 4 where we plot the performance of our hyperparameter-optimized models at different dataset sizes. Between training runs, we found the distribution of the non-negative metrics $e_{\text{pos}}$, $e_{\text{rot}}$, and $e_{\text{vel}}$ on the test set to be right skewed. We therefore assume their distributions to be log-normal, and display 95% confidence intervals calculated via Cox’s method [24]. We observe that for every metric and dataset size, prediction was harder for stiffer models. We particularly note that the Hard models perform worse when trained on 1000 example trajectories than the Soft models perform for only 100, implying a data-efficiency gap of at least 10x.

In Figure 5 we display the relationship between converged training loss $\mathcal{L}(\theta)$ and dataset size for our models. Similar to rollout error, we construct 95% log-normal confidence intervals for the mean, and additionally show confidence intervals for the coefficient of variation using the formula from [25]. Stiffer models show worse average training loss across all tested dataset sizes with high confidence. While
Fig. 5: We compare the quality of converged models across stiffness settings and dataset sizes. Estimates of the mean and coefficient of variation of test loss are plotted with 95% confidence intervals. Stiffer models show a significantly higher mean across dataset sizes. While the Hard models have a significantly larger coefficient of variation than both Soft and Medium models in the high-data regime. The large uncertainty can be attributed to the difficulty in accurately estimating population variance from small sample sizes.

there was a higher coefficient of variation between Hard models than Medium and Soft models at most dataset sizes, the difference was only statistically significant in the high-data regime.

Finally, we plot the effect of weight decay on test set loss in Figure 6. While it is true that regularization worsens performance on the test set for the Hard models, it’s not immediately clear that this effect is caused directly by stiffness. We find that the increase in test loss is consistent between stiffness settings across several orders of magnitude of regularization. We note that one fundamental purpose of regularization is to reject noise in training data, whereas our training data has none. Therefore, an appropriate future study might measure the interdependence of noise, regularization, and stiffness in test set performance.

V. DISCUSSION AND CONCLUSIONS

In this work, we have outlined a fundamental conflict between the dynamics of frictional contact and the inductive biases of common deep learning approaches. Our experiments offer compelling evidence that there are real and significant practical ramifications for this conflict. Even in a simplistic setting, our results show that both the data efficiency and the training process are significantly degraded in stiffer approximations of discontinuous contact behaviors.

While compelling, these results can only be considered as an initial study. Notably, the examined system does not depend on external inputs (actions) and has far fewer state variables than many manipulation and locomotion tasks; it is possible that the performance gap between hard and soft contact could be even wider for more complex systems. Future studies into systems with various dimensions of inputs and states will be vital in understanding this relationship. Furthermore, as the primary goal of robotic learning is to create intelligent real-world systems, experiments on real robotic systems would provide evidence that MuJoCo’s particular approximation of contact behavior is not the sole driving force in the stiffness-induced performance gap. Replicating the results of this paper with real-world materials of varying stiffness would strengthen the evidence that contact plays a role in learning performance on real systems.

Data efficiency has been a primary focus of several recent contributions to robotic learning. Approaches have focused on obtaining globally-accurate models via active curiosity-based sampling of training data to [9]; directly training on long-term prediction [15], [1], [26], [27]; and learning only locally-accurate models for completing a single task [5], [3]. While none of these methods attempt to directly handle the conflict between discontinuity and deep learning, experiments into the effects of stiffness on these algorithms would strengthen the relevance of the results presented here to the state-of-the-art.

The ultimate goal of quantifying stiffness-induced challenges is to eventually inspire algorithms which can overcome them. One possible direction is to shift deep learning’s inductive bias towards models with discontinuity instead of explicitly embedding physics priors. Methods using this strategy capture discontinuous dynamics via multi-modality
Future extensions of this work will examine residual learning. Moreover, is restricted to inelastic impact and dry friction. Capturing discontinuity, but still requires significant knowledge bias conflict and numerical condition issues by implicitly (Fig. 7).

The issue by doubling the number of discontinuities to fit discontinuities, the residual physics settings can exacerbate [1]. However, if the imperfect model does not properly place DNN that corrects an imperfect physical model [16], [15], methods rectify this issues by learning a Residual physics

same limitations as classic sysID described in Section I.

Explicitly. Learning parameters for a differentiable physics in future work. Toroidal mode explosion will be a key challenge to overcome limited to tasks with a handful of contact modes. Combining modes, whereas application of these methods has so far limited to 6 faces, 8 corners, and 12 edges; 1 for freefall), whereas application of these methods has so far limited to tasks with a handful of contact modes. Combinatorial mode explosion will be a key challenge to overcome in future work.

Another option is to embed the structure of contact explicitly. Learning parameters for a differentiable physics simulator embedded in a deep network is one previously investigated option [29], but such methods experience the same limitations as classic sysID described in Section I. Residual physics methods rectify this issues by learning a DNN that corrects an imperfect physical model [16], [15], [1]. However, if the imperfect model does not properly place discontinuities, the residual physics settings can exacerbate the issue by doubling the number of discontinuities to fit (Fig. 7).

Our previous work [14] circumvents both the inductive bias conflict and numerical condition issues by implicitly capturing discontinuity, but still requires significant knowledge of the physical properties of the system, and furthermore is restricted to inelastic impact and dry friction. Future extensions of this work will examine residual learning of the implicit discontinuity representation, allowing for incorrectly-placed discontinuities to be relocated without poor numerical conditioning.

Appendix I

1D Example

Here, we discuss the implementation of the 1D examples explored in Figures [1] and [7]. In all cases, DNNs with Tanh activations and 3 hidden layers of width 300 are fit using supervised learning and MSE loss to the data. The models in Fig. [1b] are trained directly on the data from Fig. [1]. The residual model in Fig. [7a] is trained on the red data in Fig. [7a] and the model predictions in Fig. [7b] are reconstructed by adding the network output to the yellow model from Fig. [7a]. All models are trained to convergence using Adam with minibatches of size 16 and learning rate $10^{-4}$. The residual model and the regularized model both additionally have a weight decay coefficient of $0.01$.

References


