Simultaneous Learning of Contact and Continuous Dynamics

Bibit Bianchini, Mathew Halm, and Michael Posa
GRASP Laboratory, University of Pennsylvania
{bibit, mhalm, posa}@seas.upenn.edu

Abstract: Robotic manipulation can greatly benefit from the data efficiency, robustness, and predictability of model-based methods if robots can quickly generate models of novel objects they encounter. This is especially difficult when effects like complex joint friction lack clear first-principles models and are usually ignored by physics simulators. Further, numerically-stiff contact dynamics can make common model-building approaches struggle. We propose a method to simultaneously learn contact and continuous dynamics of a novel, possibly multi-link object by observing its motion through contact-rich trajectories. We formulate a system identification process with a loss that infers unmeasured contact forces, penalizing their violation of physical constraints and laws of motion given current model parameters. Our loss is unlike prediction-based losses used in differentiable simulation. Using a new dataset of real articulated object trajectories and an existing cube toss dataset, our method outperforms differentiable simulation and end-to-end alternatives with more data efficiency. See our project page for code, datasets, and media: https://sites.google.com/view/continuous-contact-nets/home

Keywords: system identification, dynamics learning, contact-rich manipulation

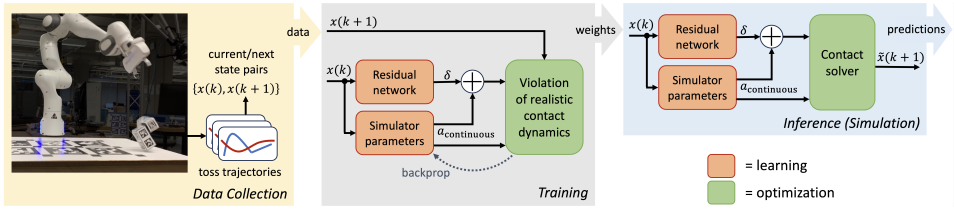

Figure 1: Our method for learning dynamics of an unknown object. **Left:** A Franka Panda automates data collection by tossing an object onto a table as the object's configuration is recorded. **Middle:** Our violation-based implicit loss without explicit simulation trains simulator parameters and a residual δ that augments the learned continuous acceleration, encouraging the residual to learn smooth accelerations characteristic of continuous dynamics, while contact-related parameters implicitly define stiffer contact dynamics. **Right:** The trained model can be used with any simulator (contact solver) during inference for performing dynamics predictions.

1 Introduction

In the future of robotic manipulation off the assembly line, robots will encounter new objects in their environment and be expected to perform useful tasks with them, such as cooking with kitchen utensils, using tools, opening doors, and packing items. Model-based control methods work increasingly well in contact-rich scenarios [1, 2], but rely on models of the manipulated objects. Unlike factory settings where everything can be precisely modeled, or locomotion where the robot itself is typically the only dynamic agent, a challenge of manipulation in the wild lies in the unknown properties of the objects to be manipulated. Model-free methods are viable, though potentially require prohibitive amounts of data [3]. Building models on the fly could enable model-based control and result in more generalizable and robust performance, but is only realistic if model-building is fast.

Manipulation is fundamentally contact-rich, and the resulting discontinuous dynamics can make model construction particularly challenging [4]. Standard system identification methods work well

7th Conference on Robot Learning (CoRL 2023), Atlanta, USA.

for identifying smooth continuous dynamics parameters even in the presence of stiff contact [5, 6], though assuming contact-related parameters are known. Pfrommer et al. [7] developed a physics-based method which circumvented numerical difficulties directly by leveraging the physical structure to derive a smooth, violation-based loss, though assuming continuous dynamics are known. It is the aim of this work to learn both continuous and contact dynamics simultaneously.

The challenge of jointly learning continuous and contact dynamics is that the overall dynamics inherit the stiffness of contact, whose impacts can overpower the smaller, smooth, albeit important continuous accelerations. Thus, separating continuous dynamics from contact events, while desirable, is not straightforward. We extend Pfrommer et al. [7] which handles the contact-related portions, then combine optimization-friendly inertial parameterizations [8] with common deep neural network (DNN) practices of encouraging smoothness to handle the continuous dynamics via residual physics, all while enjoying the data efficiency of an implicit model with suitable loss [9].

1.1 Contributions and Outline

We make the following contributions in this work:

- Present and make available a dataset of over 500 real toss trajectories of an articulated object, whose state-dependent continuous dynamics are more complicated than single rigid bodies.
- Extend prior work on learning contact dynamics [7] by simultaneously learning continuous dynamics via a combination of model-based parameterization and DNN residual physics.
- Demonstrate effective performance of our method on two real datasets and two simulation scenarios with imposed actual-to-modeled continuous dynamics gaps. We provide comparisons with differentiable simulation and end-to-end learning as alternatives.

We ground our motivations and methods in §2 with related background. §3 details model representations, followed by loss formulations in §4. With experimental setup in §5, we present and discuss the results in §6, followed by a conclusion in §7 and discussion of limitations and future work in §8.

2 Background and Related Work

Challenges of contact-rich dynamics. Detecting contact events is extremely difficult in many practical scenarios. Many model building works solve simpler variations, e.g. utilizing a contact detection oracle [10], assuming knowledge of contact distances [5, 6, 11], or operating on simple geometries like spheres or 2D interactions [6]. Our work builds off [7]'s contact-related parameter learning that performs automatic segmentation of contact/non-contact effects, without access to an oracle to identify contact events. Our work extends this contact-implicit model building beyond contact dynamics, to building full dynamics models, without impractical contact detection assumptions.

Implicit representations for discontinuous functions. Recent works have employed implicit approaches to represent sharp functions smoothly, whether those functions represent discontinuous control policies [12] or contact models [13]. Other works demonstrating differentiation through these implicit representations make them viable for use in learning [14, 15]. These techniques rely on smooth parameters to implicitly encode signals for discontinuous or extremely stiff events, which accurately characterizes contact dynamics. However, the implicit model representation can be data inefficient to optimize when combined with explicit losses [9]. Implicit model representations in combination with an informative loss can successfully learn contact parameters [7].

Differentiable simulation. Widely used for policy learning and control, differentiable simulators are also useful for system identification [5, 16, 17]. Differentiable simulators use governing dynamics that can be explicitly differentiated, and often compare simulated predictions with observed motion, using the difference to supervise model training. However because they use prediction-based losses, differentiable simulators notoriously can have difficult to optimize loss landscapes when identifying contact-related parameters [18]. Differentiable simulation can be combined with artificially soft contact models to improve optimization [19], at the cost of model accuracy [4].

Inertial parameterizations. The inertia of a rigid body is completely described by 10 parameters: the mass, center of mass (3), and moments/products of inertia (6). Learning these directly can be problematic since many members of $\mathbb{R}^{10}$ are physically infeasible inertia vectors. There are several previously developed mappings from $\theta_{\text{inertia}} \in \mathbb{R}^{10}$ to physically feasible sets of inertial parameters [8, 20, 21], and thus learning θ_{inertia} to indirectly yield inertial properties becomes a well posed optimization problem. We use the Rucker and Wensing [8] parameterization in this work.

Residual physics. While model-based structures typically boast data efficiency compared to model-free approaches [3], they fundamentally suffer from inaccuracies of the model on which they are based. Residual physics [22, 23, 24, 25, 26] mitigates this by learning an expressive residual that fills a data efficient but possibly insufficient structured model's sim-to-real gap. We use a residual physics DNN in this work to specifically augment the continuous dynamics of our structured model.

3 Model Representations

We consider a discrete dynamics model f parameterized by a set of learnable parameters θ that takes in some state $x(k)$ and set of control inputs $u(k)$ and performs a single simulation step,

$$x(k+1) = f^{\theta}(x(k), u(k)). \tag{1}$$

This makes no assumptions about the structure of f or what the learned parameters θ represent. In an unstructured case, f could be learned as a DNN where θ is the weights and biases of the network. In a more structured case (e.g. rigid body dynamics), θ represents physical parameters of the system. Numerical methods commonly simulate contact by introducing an optimization problem to search for contact impulses $\lambda(k)$ from a feasible set of contact impulses Λ over the time step,

$$x(k+1) = g^{\theta}(x(k), u(k), \lambda(k)), \tag{2a}$$

$$\text{where} \qquad \lambda(k) = \arg\min_{\lambda \in \Lambda} \; h^{\theta}\left(x(k), u(k), x(k+1), \lambda\right), \tag{2b}$$

where h^{θ} measures violation of contact constraints. This generic formulation underpins many common simulators, where the embedded optimization problem may be a linear complementarity problem (LCP) [27, 28], a second-order cone program [29], or some more generic structure [30].

3.1 Measuring Violation of Rigid Body Contact Dynamics

Inspired by the LCP formulation from Stewart and Trinkle [28], we follow standard methods for conversion to an equivalent optimization problem form in (2), introduced by Pfrommer et al. [7]. First, we let Λ describe a Coulomb friction cone,

$$\lambda \in \Lambda \quad \Leftrightarrow \quad \|\lambda_{t,i}\| \leq \mu_i \lambda_{n,i} \quad \forall i = 1, \dots p, \tag{3}$$

for a system with p contacts. Then we use penalty terms to describe violation of force complementarity, energy dissipation, and geometric penetration for each contact i,

$$h^{\theta}_{\text{comp},i}(k) = \lambda_{n,i}(k)\phi_i(k+1), \tag{4a}$$

$$h^{\theta}_{\text{diss},i}(k) = \lambda_i^T(k) \begin{bmatrix} \mu_i \left\| J_{t,i}(k) v(k+1) \right\| \\ J_{t,i}(k) v(k+1) \end{bmatrix}, \tag{4b}$$

$$h^{\theta}_{\text{pen},i}(k) = \min\left(0, \phi_i(k+1)\right)^2, \tag{4c}$$

where ϕ is the set of signed distances, μ is set of friction coefficients, $J = [J_n; J_t]$ is the normal and tangential contact Jacobians, and $x = [q; v]$ is the state of system configuration and velocity. Thus, with relative weighting between the terms, h^{θ} becomes

$$h^{\theta}(k) = \sum_{i=1}^{p} \sum_{j \in \{\text{comp,diss,pen}\}} w_j h^{\theta}_{j,i}(k). \tag{5}$$

With the contact dynamics described by h in (5) and the constraint in (3), the function g is a discretized version of Newton's third law to update system velocities, and an implicit Euler step to update the configuration. This g is a function of the implicit variable λ, and can be written as

$$v(k+1) = v(k) + a_{\text{continuous}}\Delta t + M^{-1}J^T\lambda(k), \tag{6a}$$

$$q(k+1) = q(k) + \Gamma v(k+1)\Delta t, \tag{6b}$$

where $a_{\text{continuous}}$ is the acceleration of the system due to continuous dynamics, and Γ maps velocity-space to configuration-space (e.g. mapping angular velocity to the time-derivative of an orientation quaternion). See Appendix A.1 for the selection of the introduced hyperparameter weights in h^θ.

3.2 Learnable Parameters

With the above model structure, the learnable parameters include the following. **Geometry** determines J and ϕ, both functions of the system configuration, i.e. $J(q(k)), \phi(q(k))$. We parameterize J and ϕ by a set of vertices whose 3D locations are learnable. **Friction**, via μ, determines the permissible set of contact impulses per contact point. In this work, the friction is parameterized by a single scalar μ. **Inertia** affects the model's forward predictions in (6), where M and $a_{\text{continuous}}$ appear. With articulation, M is a function of the system configuration, i.e. $M(q(k))$, and $a_{\text{continuous}}$ of the full state, $a_{\text{continuous}}(x(k))$. We map learnable parameters in $\mathbb{R}^{10}$ to a physically feasible set of 10 inertia parameters, per Rucker and Wensing [8]. Under autonomous dynamics, the mass of a system is unobservable if contact forces are not measured [6]. Thus, we keep the total system mass fixed, then learn the remaining moments and products proportionally as well as the center of mass.

3.3 Residual Network

In this work, we use a residual physics DNN to compensate for inaccuracies in the model structure in (2). Since rigid body contact solvers like [28] work reasonably well to capture real inelastic contact dynamics [31], we encourage the residual to fill gaps in the continuous dynamics. We add components to the continuous acceleration of the system,

$$a_{\text{continuous}}\left(x(k)\right) = a_{\text{continuous, model}}\left(x(k)\right) + \delta^\theta\left(x(k)\right), \tag{7}$$

where $\delta^\theta \in \mathbb{R}^{n_{\text{vel}}}$ is the output of a residual network whose input is the state of the system. See Appendix A.2 for network architecture details.

Adding costs on the norm of the network's output and on its weights encourages the residual to be small and smooth, respectively, as continuous dynamics are in comparison to contact dynamics. For end-to-end alternatives which aim to capture both continuous and contact dynamics in one network, weight regularization is no longer beneficial since it leads to unrealistically soft contact dynamics. To capture the stiffness of contact, the result is an end-to-end network with extreme input sensitivity. In contrast, incorporating our residual into the continuous acceleration allows the inherent stiffness of contact to be implicitly learned via geometric and frictional properties, leaving the residual in a smooth domain better suited for standard DNN approaches [4].

4 Loss Formulation

Our specific model structure alone does not affect the generalization capabilities of a model, and the choice of loss function is also of vital importance for stiff function classes [9]. Fig. 2 diagrammatically illustrates the differences between the losses presented in this section and how they relate to explicit versus implicit model usage for simulation.

4.1 Prediction Loss

Standard approaches in model-building or system identification [32, 17, 23] use a prediction-based loss that penalizes the error in a candidate model's predictions,

$$\mathcal{L}_{\text{prediction}} = \left\| x(k+1) - f^\theta\left(x(k), u(k)\right)\right\|^2. \tag{8}$$

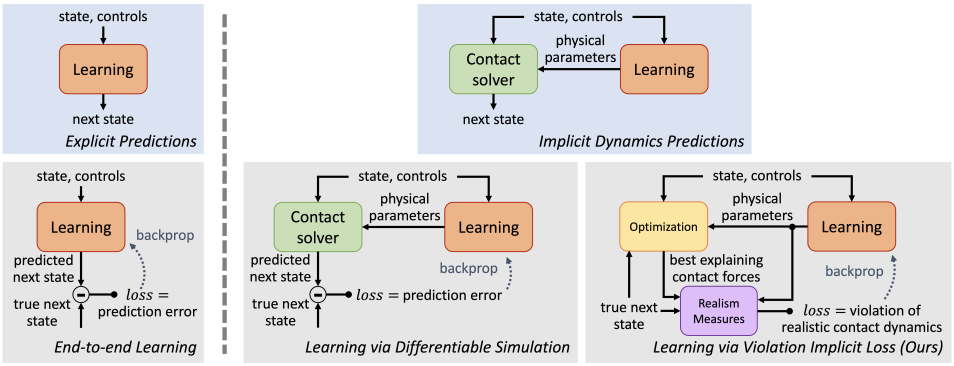

Figure 2: **Top row:** Using an explicit model (left) versus an implicit model (right) for performing dynamics predictions. Explicit models aim to predict next states directly from current states and inputs. Implicit models instead parameterize and leverage contact solvers, which produce next states as a result of an optimization problem. **Bottom row:** A common way to train an explicit model is via a prediction-based loss (left). Implicit models can also be trained with a prediction-based loss (middle), requiring performing and differentiating through the contact solver. Our approach trains an implicit model with a violation-based loss (right), avoiding simulation during training time and producing smoother, more informative gradients.

Differentiable simulators typically employ the implicit optimization problem as in (2) to solve for contact impulses, so the loss becomes (equivalently)

$$\mathcal{L}_{\text{prediction}} = \left\| x(k+1) - g^{\theta}\left(x(k), u(k), \lambda(k)\right) \right\|^2, \tag{9a}$$

$$\text{such that} \qquad \lambda(k) = \arg\min_{\lambda \in \Lambda} h^{\theta}\left(x(k), u(k), x(k+1), \lambda\right). \tag{9b}$$

Both explicit approaches (8) and implicit approaches (9) functionally result in simulating a candidate model and penalizing the difference between its prediction and the true dynamics observation.

4.2 Violation-Based Implicit Loss

Despite the increasing prevalence of implicit approaches, prediction-based losses inhibit the generalization benefits of implicit models [9]. A violation-based implicit loss of the form

$$\mathcal{L}_{\text{violation}} = \min_{\lambda \in \Lambda} \left[\left\| x(k+1) - g^{\theta}\left(x(k), u(k), \lambda\right) \right\|^2 + h^{\theta}\left(x(k), u(k), x(k+1), \lambda\right) \right], \tag{10}$$

uses h as a soft constraint and, as a result, boasts greater data efficiency [9]. This loss itself is an optimization problem that solves for the set of contact impulses λ that balances 1) explaining the observed motion and 2) matching the learned contact dynamics model. Thus minimizing this loss function through the training process is a bilevel optimization problem. For full details, see [9, 7]. This loss performs inference over contact mode, a key enabling technique for contact-implicit planning and control [1, 33].

The exact form of the prediction error term $\left\| x(k+1) - g^{\theta}\left(x(k), u(k), \lambda\right) \right\|^2$ employed herein penalizes errors in velocity space, since configurations are an affine function of velocity predictions (6b). A natural way to combine mixed linear and angular terms is to convert all into energy units via

$$l^{\theta}_{\text{pred, energy}}(k) = \left\| M \Delta v(k) + J^T \lambda \right\|^2_{M^{-1}}, \tag{11}$$

$$\text{where} \qquad \Delta v(k) = -v(k+1) + v(k) + a_{\text{continuous}} \Delta t. \tag{12}$$

5 Experimental Setup

For all experiments, we consider a system autonomously falling under gravity and colliding with a flat plane. In addition to the **cube toss** dataset contributed by Pfrommer et al. [7], we contribute one new real dataset and two simulated scenarios. We used a Franka Emika Panda 7 degree-of-freedom robotic arm to automate the toss data collection of a **two-link articulated object**. Pose information

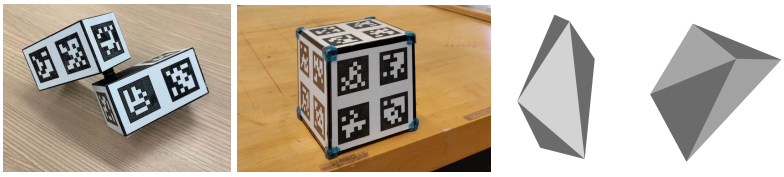

Figure 3: Our experimental systems. **Left:** A real two-link articulated object, with each link as 5cm by 5cm by 10cm. **Left middle:** A real cube of width 10cm, whose dataset was contributed by [7]. **Right:** A simulated 6-vertex asymmetric object from two views. The volume of the asymmetric is similar to the volume of one of the articulated object links.

of each link is tracked using TagSLAM [34]. These two body poses are combined into minimal coordinates via an optimization problem that minimizes pose offset of both links. While we keep the system mass fixed, our model can freely decide the mass distribution across the two links.

We add a **vortex simulation of an asymmetric object** example, simulating dynamics with a spatially-varying force field pulling towards and swirling around a fixed vertical line. The initialized model is unaware of this continuous dynamics augmentation. We test in this scenario with an asymmetric object with 6 vertices. Lastly, we include a **gravity simulation of the articulated object**. This scenario features simulated dynamics of the articulated object from our new dataset with typical gravitational acceleration of $9.81\text{m}/\text{s}^2$. We test at a fixed training set size of 256 tosses from poor initial guesses, at some fraction $\in [0, 2]$ of this simulated gravity. All simulation data was generated using Drake [35] and features a significant gap between the model's believed and the simulator's actual dynamics. See Fig. 3 for visuals of these systems.

Parameter Initializations. All experiments were run a minimum of 9 times each with a random parameter initialization using a process described in Appendix A.4. Shaded regions in the results plots indicate 5%/95% normal t-score confidence intervals.

Comparisons and Evaluation Metrics. See Table 1 for the five approaches we tested. Anitescu dynamics [36] were selected as a reasonable differentiable simulation baseline, as it forms the basis of many widely-used, modern simulators, notably including MuJoCo [30] and Drake [35]. We present prediction errors for all approaches and parameter errors for the structured approaches. Prediction errors are the average norm error of all bodies' position or orientation over the course of a trajectory. Defining $\mathcal{V}$ as the set of points inside a body's geometry and $\mathcal{I} = [m, p_x, p_y, p_z, I_{xx}, I_{yy}, I_{zz}, I_{xy}, I_{xz}, I_{yz}]$ as the set of body inertial parameters, parameter errors for quantifying the geometry, friction, and inertial properties for a system with n bodies are

$$e_{\text{volume}} = \frac{1}{n}\sum_{i=1}^{n} \frac{\text{Vol}\left(\left(\mathcal{V}_{i,\text{actual}} \setminus \mathcal{V}_{i,\text{learned}}\right) \cup \left(\mathcal{V}_{i,\text{learned}} \setminus \mathcal{V}_{i,\text{actual}}\right)\right)}{\text{Vol}\left(\mathcal{V}_{i,\text{actual}}\right)}, \tag{13a}$$

$$e_{\text{friction}} = \left\| [\mu_1, \ldots, \mu_n]_{\text{learned}} - [\mu_1, \ldots, \mu_n]_{\text{actual}} \right\|, \tag{13b}$$

$$e_{\text{inertia}} = \left\| [s \cdot \mathcal{I}_1, \ldots, s \cdot \mathcal{I}_n]_{\text{learned}} - [s \cdot \mathcal{I}_1, \ldots, s \cdot \mathcal{I}_n]_{\text{actual}} \right\|. \tag{13c}$$

The vector s is akin to a "characteristic length" that is effectively normalized by the inertia of the true object. See more details on the volume and inertia metrics in Appendices B.1 and B.2, respectively.

Name	Parameterization	Loss	Residual
CCN (ours)	Structured	Violation implicit	
CCN-R (ours)	Structured	Violation implicit	✓
DiffSim	Structured	Prediction error	
DiffSim-R	Structured	Prediction error	✓
End-to-end	DNN	Prediction error	N/A

Table 1: Tested approaches. CCN stands for our extension of Continuous dynamics learning plus the contact dynamics learning in ContactNets [7]. DiffSim is Differentiable Simulation using differentiable contact dynamics defined in Anitescu [36]. The -R modifier indicates residual physics is included. DiffSim ablates our violation implicit loss function, and the End-to-end baseline ablates the physical structure imposed by the rigid body model-based parameterization. See Appendix A.3 for End-to-end network details.

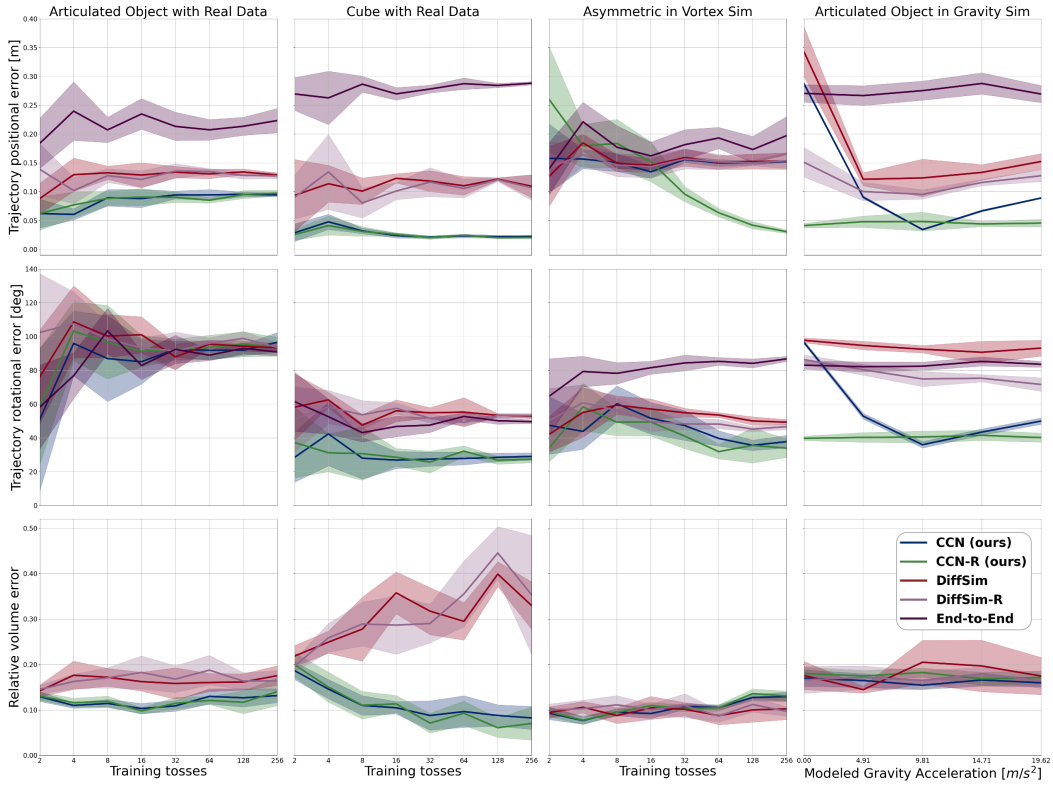

Figure 4: Results from the four experiments. Shaded regions indicate normal t-score 95% confidence intervals. **Left column:** The real articulated object featured rotations that were difficult for any of the methods to capture over a long time horizon (middle), yet our CCN approaches outperforms DiffSim on geometry error (bottom) and all alternatives on positional error (top). **Left middle column:** For every metric on the real cube experiments, our CCN approaches outperform DiffSim and End-to-end. **Right middle column:** While every method achieved low geometry error on the asymmetric object in simulated vortex dynamics, our CCN approaches performed the best in rotational error, and only our approach with residual (CCN-R) was able to achieve low positional error. **Right column:** The x-axis for the gravity experiments swept over an initial modeled gravitational acceleration. Despite poor model discrepancy, only our approach with residual CCN-R is able to maintain good performance across all metrics at different model discrepancies.

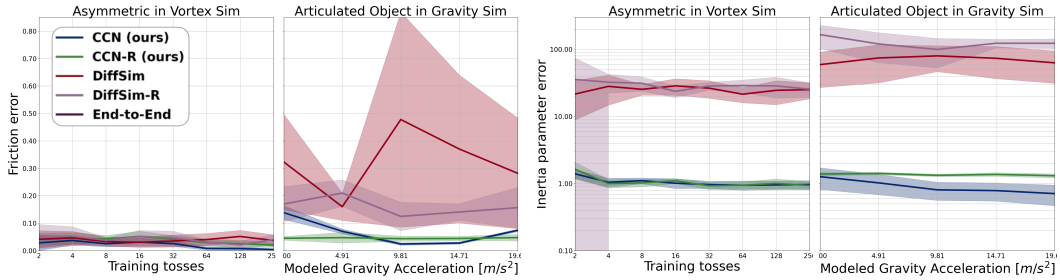

Figure 5: Friction (**Left**, **Left middle**) and inertia (**Right middle**, **Right**) parameter errors for the vortex and gravity simulated experiments.

6 Results

We test the methods across challenging datasets featuring collisions through contact-rich trajectories. While the contact dynamics are prominent in all the example trajectories, we built the articulated system in particular for its continuous dynamics: non-trivial due to state-dependent Coriolis and centrifugal effects and unmodeled joint friction, damping, or backlash.

Observations. Both real experiments (articulated object and cube in left and left middle columns in Fig. 4, respectively) show separation between CCN, DiffSim, and End-to-end methods, with CCN matching or outperforming alternatives along all metrics, especially with more data. On the cube dataset, CCN and CCN-R consistently converge to $<10\%$ volume error while DiffSim struggles to improve even with more data. The residual does not significantly help with real data but does in simulated examples, where CCN-R in the vortex scenario (right middle column in Fig. 4) improves its positional trajectory error significantly, achieving consistently 5x better performance than other methods at the largest dataset size. On the same metric, DiffSim-R sees no improvement beyond DiffSim. Fig. 5 shows CCN and CCN-R nearly always outperforms DiffSim and DiffSim-R, except for the vortex scenario where all methods perform well. In the gravity scenario (right column in Fig. 4), the residual helps CCN-R maintain good performance achieved at the correct gravitational acceleration, across all initial models. In contrast, DiffSim-R outperforms DiffSim at every initial gravitational acceleration model. Since this gravity scenario swept over different modeled gravitational accelerations, End-to-end is unaffected since its representation is unstructured, and its consistent performance over the x-axis of these plots are included for reference against CCN and DiffSim. Parameter errors in Fig. 5 indicate DiffSim and DiffSim-R struggled to capture both the friction and inertial terms in the gravity scenario to levels attained by CCN and CCN-R.

Implications. The residual's lack of effect on real data is in alignment with prior works that found the rigid body model already performs well in contact-rich scenarios with simple systems [31]. The residual shows the most merit in the more extreme simulation examples, though its effect isn't realized until larger dataset sizes – unsurprising for a DNN. The residual helps DiffSim to a much lesser extent because our method better separates continuous and contact dynamics and allows the residual to identify the smooth nature of the unmodeled vortex dynamics. Relatedly, there is better performance for DiffSim and DiffSim-R at overestimated gravitational accelerations rather than underestimated, where contact is less often predicted. Without contact, prediction losses experience a lack of informative parameter gradients, in which case DiffSim-R outperforms DiffSim.

7 Conclusion

We demonstrate with real experiments that our violation implicit loss trains models that outperform prediction loss-based structured and unstructured models. Our approach leverages the structure of contact versus continuous dynamics to learn both simultaneously, with physically meaningful parameters driving separate contact and continuous dynamics with a DNN residual to augment.

8 Limitations and Future Work.

The articulated object is the most challenging system presented herein for dynamics learning. While our methods outperformed alternatives in all other metrics, there is still a significant gap between ground truth and our models' trajectory predictions, and the rotational error showed lackluster performance from all methods. Further closing this gap remains for future exploration, and we are open-sourcing our articulated object dataset for the community to contribute their own methods. The scalability of the method in this paper has not yet been demonstrated on large-scale systems (e.g. a robotic arm) or in multi-object settings. It remains to be seen whether the advantages demonstrated here will extend as scope increases. While we tested one version of differentiable simulation using Anitescu [36] dynamics, future studies will compare alternatives against each other and our violation implicit loss in performing system identification. Our approach encouraged the residual to fill gaps in continuous dynamics while relying on a rigid body contact dynamics model to handle contact. Other works have demonstrated improved prediction capability by learning the contact model [37], though integrating this with system identification remains future work. While other works have learned articulated structures from scratch [38, 39, 40], we assumed access to kinematic structure in this paper, leaving joint kinematics/dynamics learning for future studies. Lastly, we relied on AprilTags to estimate poses, which are more challenging to obtain via perception [41, 42, 43].

Acknowledgments

We thank our anonymous reviewers, who provided thorough and fair feedback. This work was supported by a National Defense Science and Engineering Graduate Fellowship, an NSF Graduate Research Fellowship under Grant No. DGE-1845298, and an NSF CAREER Award under Grant No. FRR-2238480.

References

[1] A. Aydinoglu, A. Wei, and M. Posa. Consensus complementarity control for multi-contact mpc. *arXiv preprint arXiv:2304.11259*, Apr. 2023.

[2] A. Aydinoglu and M. Posa. Real-time multi-contact model predictive control via admm. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 3414–3421. IEEE, 2022.

[3] L. Ljung. Perspectives on system identification. *Annual Reviews in Control*, 34(1):1–12, 2010.

[4] M. Parmar, M. Halm, and M. Posa. Fundamental challenges in deep learning for stiff contact dynamics. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5181–5188. IEEE, 2021.

[5] F. de Avila Belbute-Peres, K. Smith, K. Allen, J. Tenenbaum, and J. Z. Kolter. End-to-end differentiable physics for learning and control. *Advances in neural information processing systems*, 31:7178–7189, 2018.

[6] N. Fazeli, R. Kolbert, R. Tedrake, and A. Rodriguez. Parameter and contact force estimation of planar rigid-bodies undergoing frictional contact. *The International Journal of Robotics Research*, 36(13-14):1437–1454, 2017.

[7] S. Pfrommer, M. Halm, and M. Posa. ContactNets: Learning Discontinuous Contact Dynamics with Smooth, Implicit Representations. In *The Conference on Robot Learning (CoRL)*, 2020. URL https://proceedings.mlr.press/v155/pfrommer21a.html.

[8] C. Rucker and P. M. Wensing. Smooth parameterization of rigid-body inertia. *IEEE Robotics and Automation Letters*, 7(2):2771–2778, 2022.

[9] B. Bianchini, M. Halm, N. Matni, and M. Posa. Generalization bounded implicit learning of nearly discontinuous functions. In *Learning for Dynamics and Control Conference*, pages 1112–1124. PMLR, 2022.

[10] A. Hochlehnert, A. Terenin, S. Sæmundsson, and M. Deisenroth. Learning contact dynamics using physically structured neural networks. In *International Conference on Artificial Intelligence and Statistics*, pages 2152–2160. PMLR, 2021.

[11] K. R. Allen, Y. Rubanova, T. Lopez-Guevara, W. Whitney, A. Sanchez-Gonzalez, P. Battaglia, and T. Pfaff. Learning rigid dynamics with face interaction graph networks. *arXiv preprint arXiv:2212.03574*, 2022.

[12] P. Florence, C. Lynch, A. Zeng, O. A. Ramirez, A. Wahid, L. Downs, A. Wong, J. Lee, I. Mordatch, and J. Tompson. Implicit behavioral cloning. In *Conference on Robot Learning*, pages 158–168. PMLR, 2022.

[13] N. Fazeli, S. Zapolsky, E. Drumwright, and A. Rodriguez. Learning data-efficient rigid-body contact models: Case study of planar impact. In *Conference on Robot Learning*, pages 388–397. PMLR, 2017.

[14] A. Agrawal, S. Barratt, S. Boyd, E. Busseti, and W. M. Moursi. Differentiating through a cone program. *arXiv preprint arXiv:1904.09043*, 2019.

[15] B. Amos and J. Z. Kolter. Optnet: Differentiable optimization as a layer in neural networks. In *International Conference on Machine Learning*, pages 136–145. PMLR, 2017.

[16] S. Le Cleac'h, M. Schwager, Z. Manchester, V. Sindhwani, P. Florence, and S. Singh. Single-level differentiable contact simulation. *IEEE Robotics and Automation Letters*, 2023.

[17] T. A. Howell, S. L. Cleac'h, J. Z. Kolter, M. Schwager, and Z. Manchester. Dojo: A differentiable simulator for robotics. *arXiv preprint arXiv:2203.00806*, 2022.

[18] R. Antonova, J. Yang, K. M. Jatavallabhula, and J. Bohg. Rethinking optimization with differentiable simulation from a global perspective. In *Conference on Robot Learning*, pages 276–286. PMLR, 2023.

[19] R. Tedrake. *Underactuated Robotics*. 2023. URL https://underactuated.csail.mit.edu.

[20] G. Sutanto, A. Wang, Y. Lin, M. Mukadam, G. Sukhatme, A. Rai, and F. Meier. Encoding physical constraints in differentiable newton-euler algorithm. In *Learning for Dynamics and Control*, pages 804–813. PMLR, 2020.

[21] 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*, 5(3):101–119, 1986.

[22] J. Wong, V. Makoviychuk, A. Anandkumar, and Y. Zhu. Oscar: Data-driven operational space control for adaptive and robust robot manipulation. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 10519–10526. IEEE, 2022.

[23] E. Heiden, D. Millard, E. Coumans, Y. Sheng, and G. S. Sukhatme. Neuralsim: Augmenting differentiable simulators with neural networks. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9474–9481. IEEE, 2021.

[24] P. Abbeel, M. Quigley, and A. Y. Ng. Using inaccurate models in reinforcement learning. In *Proceedings of the 23rd international conference on Machine learning*, pages 1–8, 2006.

[25] A. Ajay, J. Wu, N. Fazeli, M. Bauza, L. P. Kaelbling, J. B. Tenenbaum, and A. Rodriguez. Augmenting physical simulators with stochastic neural networks: Case study of planar pushing and bouncing. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3066–3073. IEEE, 2018.

[26] A. Zeng, S. Song, J. Lee, A. Rodriguez, and T. Funkhouser. Tossingbot: Learning to throw arbitrary objects with residual physics. *IEEE Transactions on Robotics*, 36(4):1307–1319, 2020.

[27] M. Anitescu and F. A. Potra. Formulating dynamic multi-rigid-body contact problems with friction as solvable linear complementarity problems. *Nonlinear Dynamics*, 14(3):231–247, 1997.

[28] D. E. Stewart and J. C. Trinkle. An implicit time-stepping scheme for rigid body dynamics with inelastic collisions and coulomb friction. *International Journal for Numerical Methods in Engineering*, 39(15):2673–2691, 1996.

[29] A. M. Castro, F. N. Permenter, and X. Han. An unconstrained convex formulation of compliant contact. *IEEE Transactions on Robotics*, 2022.

[30] E. Todorov, T. Erez, and Y. Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033. IEEE, 2012.

[31] B. Acosta, W. Yang, and M. Posa. Validating robotics simulators on real-world impacts. *IEEE Robotics and Automation Letters*, 7(3):6471–6478, 2022.

[32] A. Nagabandi, K. Konolige, S. Levine, and V. Kumar. Deep dynamics models for learning dexterous manipulation. In *Conference on Robot Learning*, pages 1101–1112. PMLR, 2020.

[33] M. Posa, C. Cantu, and R. Tedrake. A direct method for trajectory optimization of rigid bodies through contact. *The International Journal of Robotics Research*, 33(1):69–81, 2014.

[34] B. Pfrommer and K. Daniilidis. Tagslam: Robust slam with fiducial markers. *arXiv preprint arXiv:1910.00679*, 2019.

[35] R. Tedrake and the Drake Development Team. Drake: Model-based design and verification for robotics, 2019. URL https://drake.mit.edu.

[36] M. Anitescu. Optimization-based simulation of nonsmooth rigid multibody dynamics. *Mathematical Programming*, 105:113–143, 2006.

[37] K. R. Allen, T. L. Guevara, Y. Rubanova, K. Stachenfeld, A. Sanchez-Gonzalez, P. Battaglia, and T. Pfaff. Graph network simulators can learn discontinuous, rigid contact dynamics. In *Conference on Robot Learning*, pages 1157–1167. PMLR, 2023.

[38] J. Sturm, C. Stachniss, and W. Burgard. A probabilistic framework for learning kinematic models of articulated objects. *Journal of Artificial Intelligence Research*, 41:477–526, 2011.

[39] L. Ma, J. Meng, S. Liu, W. Chen, J. Xu, and R. Chen. Sim2real2: Actively building explicit physics model for precise articulated object manipulation. *arXiv preprint arXiv:2302.10693*, 2023.

[40] N. Heppert, M. Z. Irshad, S. Zakharov, K. Liu, R. A. Ambrus, J. Bohg, A. Valada, and T. Kollar. Carto: Category and joint agnostic reconstruction of articulated objects. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21201–21210, 2023.

[41] B. Wen and K. Bekris. Bundletrack: 6d pose tracking for novel objects without instance or category-level 3d models. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 8067–8074. IEEE, 2021.

[42] H. Chen, F. Manhardt, N. Navab, and B. Busam. Texpose: Neural texture learning for self-supervised 6d object pose estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4841–4852, 2023.

[43] Y. Liu, Y. Wen, S. Peng, C. Lin, X. Long, T. Komura, and W. Wang. Gen6d: Generalizable model-free 6-dof object pose estimation from rgb images. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXXII*, pages 298–315. Springer, 2022.

A Learning Details

Hyperparameter	CCN, real	CCN, sim	DiffSim, real	DiffSim, sim
w_{comp}	0.001	0.001	N/A	N/A
w_{diss}	0.1	0.1	N/A	N/A
w_{pen}	100	100	N/A	N/A
w_{res}	1	0.001	1000	1
$w_{\text{res, w}}$	0.1	0	1	0

Table 2: Tuned hyperparameters. Rows for residual norm (w_{res}) and weight ($w_{\text{res, w}}$) regularization only apply for -R variations. Real versus simulated experiments performed best with different residual regularization weights since the simulations featured larger model-to-actual dynamics gaps.

A.1 Model-Based Parameter Learning

For our CCN and CCN-R method, we performed a hyperparameter search to determine the most effective set of weights for balancing the loss terms in (5). See Table 2 for these sets of weights.

A.2 Residual Network Architecture and Regularization

The residual network featured in both CCN-R and DiffSim-R has the same architecture. The first layer takes in the full state of the system and converts the quaternion orientation representation into a 9-vector of the elements of the corresponding rotation matrix, letting the remaining state positions and velocities pass through to layer 2. Beyond the first layer, the network is a fully-connected multi-layer perceptron (MLP) with two hidden layers of size 128. The last layer outputs values in the acceleration space of the system. All activations are ReLU.

We regularized the residual via both output norm regularization and weight regularization, with associated weight hyperparameters w_{res} and $w_{\text{res, w}}$, respectively. See Table 2 for the optimal values. Since the simulation examples were specifically designed to test the capabilities of the residual network, we found the optimal weights for the residual terms were much lower for simulated examples than for the real data. We also note that the optimal residual weights were much higher for DiffSim than for CCN. This is a direct result from the DiffSim residual's attempts to explain some of the contact dynamics, whose accelerations are orders of magnitude larger than the continuous accelerations. Our CCN method avoids this by better containing its residual in the continuous domain, and thus could use lower residual regularization weights.

A.3 End-to-End Network Architecture

The best performing network for the End-to-end baseline is an MLP with 4 hidden layers each of size 256 with Tanh activation. Its input is the full state of the system, and its output is the next velocity. The next configuration is obtained from predicted next velocity with an Euler step (6b).

A.4 Parameter Initializations

All learned parameters are randomly initialized within pre-specified ranges. Geometric parameters are initialized between 0.5 and 1.5 times their true lengths, and friction coefficients between 0.5 and 1.5 times their approximate true values. Inertial parameters are initialized to a set of physically feasible values via the following procedure for each link in the body:

1. A virtual link is sized via a random set of three length scales l_x, l_y, l_z, chosen between 0.5 and 1.5 times the link's true dimensions.
2. The center of mass of the link is initialized to be somewhere within the inner half of this virtual link's geometry.
3. A random mass m_{rand} is selected from the range between $0.5m_{\text{urdf}}$ and $1.5m_{\text{urdf}}$.

4. Principal axis moments of inertia are computed using the assumption of uniform density throughout the randomly-sized virtual link, via e.g. for I_{xx}:

$$I_{xx,\text{principal axis}} = \frac{m_{\text{rand}}}{12}\left(l_y^2 + l_z^2\right). \tag{14}$$

5. Rotate the inertia matrix from its principal axis definition by a random rotation in SO(3). The link's initialized moments and products of inertia are derived from this rotated version.

B Evaluation Metric Details

B.1 Volume Evaluation Metric

The volume error metric defined in Equation (13a) is computed as the fraction of volume that the learned geometry incorrectly included or incorrectly excluded. To compute this, we used the identity

$$\text{Vol}(A \setminus B) = \text{Vol}(A) - \text{Vol}(A \cap B). \tag{15}$$

The numerator of (13a) can therefore be computed as

$$\text{Vol}(\mathcal{V}_{i,\text{actual}}) + \text{Vol}(\mathcal{V}_{i,\text{learned}}) - 2\text{Vol}(\mathcal{V}_{i,\text{actual}} \cap \mathcal{V}_{i,\text{learned}}). \tag{16}$$

$\mathcal{V}_{i,\text{learned}}$, $\mathcal{V}_{i,\text{actual}}$, and their intersection are all convex hulls of a finite number of vertices. Therefore, the interaction operation as well as volume calculation can be conducted with a standard convex hull or halfspace intersection library, such as `qhull`.

B.2 Inertia Evaluation Metric

A body's set of inertial parameters is $\mathcal{I} = [m, p_x, p_y, p_z, I_{xx}, I_{yy}, I_{zz}, I_{xy}, I_{xz}, I_{yz}]$. Since true inertia parameter vectors feature values at wildly different scales, the vector s is selected to normalize $\mathcal{I}$ to more equally evaluate all inertial parameter errors. For example, the true inertial parameters for the simulated asymmetric object used in the vortex example are

$$\mathcal{I}_{\text{asym}} = [0.25, 0, 0, 0, 0.00081, 0.00081, 0.00081, 0, 0, 0]. \tag{17}$$

Choosing 3.5cm as a reasonable center of mass location distance, the associated s_{asym} normalizer is

$$s_{\text{asym}} = \left[\frac{1}{0.25}, \frac{1}{0.035}, \frac{1}{0.035}, \frac{1}{0.035}, \frac{1}{0.00081}, \frac{1}{0.00081}, \frac{1}{0.00081}, \frac{1}{0.00081}, \frac{1}{0.00081}, \frac{1}{0.00081}\right]. \tag{18}$$