Proceedings of Machine Learning Research vol 211:1–14, 2023 5th Annual Conference on Learning for Dynamics and Control

Improving Gradient Computation for Differentiable Physics Simulation with Contacts

Yaofeng Desmond Zhong YAOFENG.ZHONG@SIEMENS.COM
Siemens Corporation, Technology

Jiequn Han JIEQUNHAN@GMAIL.COM
Flatiron Institute

Biswadip Dey BISWADIP.DEY@SIEMENS.COM
Siemens Corporation, Technology

Georgia Olympia Brikis GEORGIA.BRIKIS@SIEMENS.COM
Siemens Corporation, Technology

Editors: N. Matni, M. Morari, G. J. Pappas

Abstract

Differentiable simulation enables gradients to be back-propagated through physics simulations. In this way, one can learn the dynamics and properties of a physics system by gradient-based optimization or embed the whole differentiable simulation as a layer in a deep learning model for downstream tasks, such as planning and control. However, differentiable simulation at its current stage is not perfect and might provide wrong gradients that deteriorate its performance in learning tasks. In this paper, we study differentiable rigid-body simulation with contacts. We find that existing differentiable simulation methods provide inaccurate gradients when the contact normal direction is not fixed - a general situation when the contacts are between two moving objects. We propose to improve gradient computation by continuous collision detection and leverage the time-of-impact (TOI) to calculate the post-collision velocities. We demonstrate our proposed method, referred to as TOI-Velocity, on two optimal control problems. We show that with TOI-Velocity, we are able to learn an optimal control sequence that matches the analytical solution, while without TOI-Velocity, existing differentiable simulation methods fail to do so.

Keywords: differentiable simulation, rigid-body simulation, collision and contacts, optimal control

1. Introduction

With rapid advances and development of machine learning and automatic differentiation tools, a family of techniques has emerged to make physics simulation end-to-end differentiable (Liang and Lin, 2020). These differentiable physics simulators make it easy to use gradient-based methods for learning and control tasks, such as system identification (Zhong et al., 2021; Le Lidec et al., 2021; Song and Boularias, 2020a), learning to slide unknown objects (Song and Boularias, 2020b), shape optimization (Strecke and Stueckler, 2021; Xu et al., 2021) and grasp synthesis (Turpin et al., 2022). These applications demonstrate the potential of differentiable simulations in solving control and design problems that are hard to solve with traditional tools. Compared to black-box neural network counterparts, differentiable simulations utilize physical models to provide more reliable gradient information and better interpretability, which benefits various learning tasks involving physics simulations. One important and popular category of differentiable simulation investigates rigid-body simulation with collisions and contacts. However, current methods might compute gradients incorrectly, providing useless or even harmful signals for learning and optimization tasks. In the present study, we directly identify why wrong gradients occur in the original optimization problem and propose a novel technique to improve gradient computation. Our results on two optimal control examples clearly show the advantage of our proposed method. It is worth noting that another line of research in recent literature has attempted to address the challenge by modifying the optimization problem by incorporating randomness into the objective function (Suh et al., 2022b,a; Lidec et al., 2022) or implementing a smooth critic (Xu et al., 2022).

2. Preliminaries

2.1. Differentiable Simulation with Contacts

In this section, we provide a brief overview of the different types of differentiable contact models. We classify these methods into the following two categories.

Velocity-impulse-based methods treat contact events as instantaneous velocity changes. There are many ways to solve these velocity impulses. For frictional contacts, the problem can be formulated as a nonlinear complementarity problem (NCP) (Howell et al., 2022). Most existing works approximate the NCP by a *linear complementarity problem (LCP)* and apply different techniques to calculate the gradients (de Avila Belbute-Peres et al., 2018; Heiden et al., 2021b; Degrave et al., 2019; Qiao et al., 2021; Werling et al., 2021; Du et al., 2021; Li et al., 2021). Another line of research solves velocity impulses by formulating it as a *convex optimization problem* (Todorov, 2011; Todorov et al., 2012; Todorov, 2014). Zhong et al. (2021) implement a differentiable version of it with CvxpyLayer (Agrawal et al., 2019). If the contact is frictionless, the velocity impulses can be computed directly in a straightforward way, e.g., as done in Chen et al. (2021), and we refer to it as the *direct velocity impulse* method.

Non-velocity-impulse-based methods treat contact events in different ways. *Compliant models* resolve contact in multiple consecutive time steps and are studied extensively in the context of differentiable simulation (Carpentier and Mansard, 2018; Xu et al., 2022; Heiden et al., 2021b; Murthy et al., 2021; Geilinger et al., 2020; Li et al., 2020; Heiden et al., 2021a; Du et al., 2021; Macklin, 2022; Geilinger et al., 2020). Besides compliant models, *position-based dynamics (PBD)* (Müller et al., 2007; Macklin et al., 2016) that directly manipulate positions to resolve contacts can also be easily made differentiable as done in Macklin (2022); Freeman et al. (2021); Macklin et al. (2020); Liang et al. (2019); Qiao et al. (2020); Yang et al. (2020); Hu et al. (2020).

We refer readers to Zhong et al. (2022) for a more detailed overview of these methods.

2.2. Time-of-impact - Position

Hu et al. (2020) is among the first to study differentiable simulation with contacts from the perspective of velocity impulses. They find that the most straightforward implementation might produce wrong gradients due to time discretization and they propose to use continuous collision detection to find the time-of-impact (TOI) to improve gradient computation. We refer to it as TOI-Position since it leverages TOI to adjust the post-collision position. In this paper, we argue that TOI-Position is not enough to solve all the issues caused by time discretization, especially if the contact normal is not fixed over the optimization iterations. We propose a new technique for velocity-impulse-based methods to improve gradient computation.

3. Motivating Problem

In this section, we revisit an optimal control problem studied by Hu et al. (2022) and demonstrate that current velocity-impulse-based differentiable simulation methods cannot learn an optimal control input for this problem.

3.1. Problem Setup

The system is shown in Figure 1. The two blue circles represent the initial position of two balls, respectively. The pre-collision trajectory of Ball 1 is shown in green, and the post-collision trajectory of Ball 2 is shown in red. The goal is to push Ball 1 to strike Ball 2 so that Ball 2 will reach the target at the end of the simulation. We will formulate it as an optimal control problem later in Section 5. Hu et al. (2022) uses the hybrid minimum principle (HMP) to obtain an analytical solution to this class of optimal control problems. We will use this analytical solution to evaluate the performance of different methods.

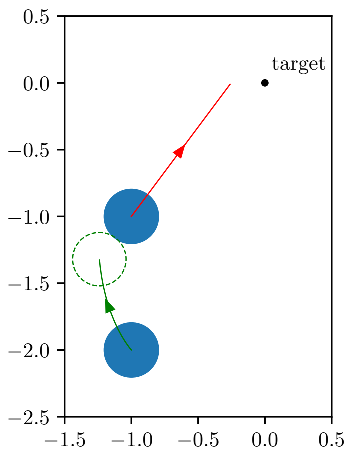

Figure 1: Motivating problem

3.2. Learning with Direct Velocity Impulse

An optimal control problem can be viewed as a constrained optimization problem where we design the control input at each time step to minimize an objective function. With differentiable simulation, we can compute the gradients of the objective with respect to the control inputs and use gradient-based optimization approaches to learn an optimal control sequence that minimizes the objective/loss function. Figure 2 shows the learning curves of the direct velocity impulse method implemented in Taichi and PyTorch, along with the analytically obtained optimal value. We observe that both implementations fail to converge to the analytical value even when TOI-Position is used. In Section 5, we will see that the learned control inputs do not match the analytical solution either (Figure 5). This result highlights the issues present in existing differentiable simulation methods.

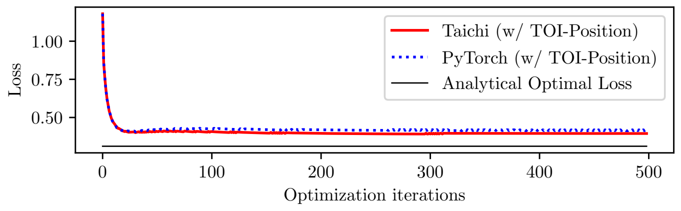

Figure 2: Learning curves of direct velocity impulse method implemented in Taichi and PyTorch.

3.3. A Hint of the Issue

To understand why the learning curve does not converge to the analytically obtained optimal solution, we initiate the learning with the analytical solution. Figure 3 shows the learning curve over the first 100 iterations. We observe that the loss jumps up in certain iterations, and the learning converges to a solution with a higher loss. This behavior is persistent across different learning rates, which indicates that the gradients computed by the differentiable simulation are incorrect.

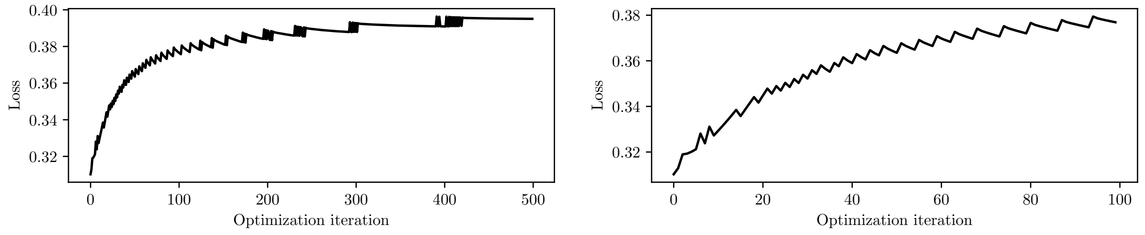

Figure 3: Motivating problem: Learning curve initialized from the analytical optimal solution.

4. Method

In this section, we first explain why the loss function increases in Figure 3 and then introduce a new technique, referred to as *TOI-Velocity*, to improve gradient calculation in differentiable simulation. We mainly work with a discrete-time formulation, where the simulation duration T is discretized into N time steps with $\Delta t = T/N$. As the examples in the paper involve two objects moving in the 2D space, we use $p_n = [p_{1,n}, p_{2,n}] = [p_{x1,n}, p_{y1,n}, p_{x2,n}, p_{y2,n}]$ to denote the x and y coordinates of the two objects at time step n. When we need to distinguish variables from different optimization iterations, we use the superscript i to denote variables in iteration i.

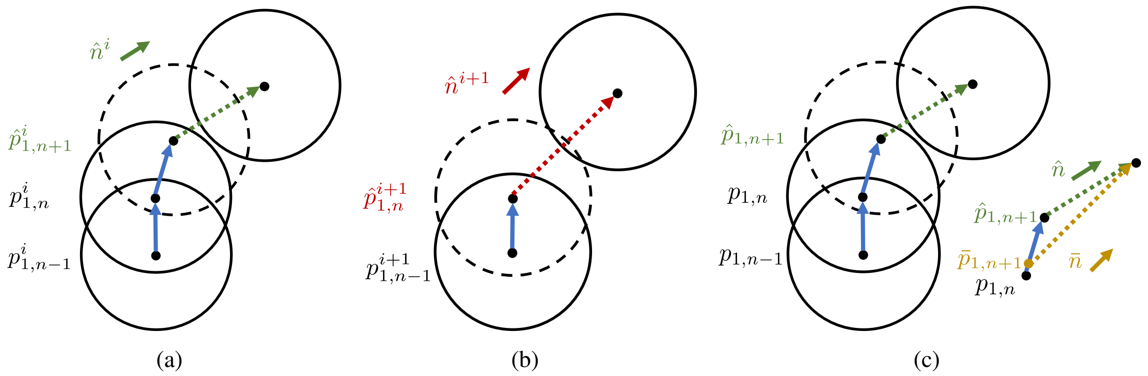

Figure 4: (a) ball positions in iteration i ; (b) ball positions in iteration $i+1$; (c) Difference between penetration direction $\hat{n}$ and collision direction $\bar{n}$ in an arbitrary iteration.

4.1. The Reason Behind Loss Increase

After taking a detailed look, we find that the increase of loss appears when the time step at which the collision happens changes over optimization iterations. The increase in the loss can be explained using the diagrams in Figure 4. In Figure 4(a), $p^i_{1,n-1}$ and $p^i_{1,n}$ show the position of the Ball 1 at time step $n-1$ and n in iteration i, respectively. $\hat{p}^i_{1,n+1}$ denotes the penetrated position of Ball 1, which is an intermediate variable used to resolve the collision (explained below). The direction of the post-collision velocity of Ball 2 is determined by the penetration direction, indicated by the green arrow. Assume that after a gradient update, the position of the balls changes to the one shown in Figure 4(b), where the collision happens in time step n instead of time step $n+1$. Now the direction of the post-collision velocity of Ball 2 is determined by the penetration direction indicated by the red arrow. As the change of penetration direction is not continuous, the change in the post-collision velocity of Ball 2 is also not continuous. Thus, the final position of Ball 2 suffers from a sudden change over these iterations, which could cause an increase in the loss since the terminal cost in the objective depends on the final position of Ball 2. This discontinuity in velocity between consecutive gradient updates is the main reason of the loss increase in Figure 3.

4.2. Time-of-impact - Velocity

To improve gradient computation in aforementioned situations where contact normal directions are not fixed, we propose to adjust post-collision velocity by continuous contact detection, and we call the technique TOI-Velocity. We present TOI-Velocity using the two-ball collision scenario shown in Figure 4(c), but the idea applies to collisions among multiple objects with a general class of shapes.

We assume we are using the symplectic Euler integration scheme, but the idea applies to other integration schemes as well. With symplectic Euler, we have

$$\hat{v}_{n+1} = v_n + u_n \Delta t, \tag{1}$$
$$\hat{p}_{n+1} = p_n + \hat{v}_{n+1} \Delta t, \tag{2}$$

where $v_n = [v_{1,n}, v_{2,n}] = [v_{x1,n}, v_{y1,n}, v_{x2,n}, v_{y2,n}]$ is the velocity of two balls at the nth time step and $u_n = [u_{1,n}, u_{2,n}] = [u_{x1,n}, u_{y1,n}, u_{x2,n}, u_{y2,n}]$ is the control input to the two balls at the nth time step. Variables with a hat $(\hat{\cdot})$ denote intermediate variables before a collision is resolved. If there is no collision detected in this time step, we would have $v_{n+1} = \hat{v}_{n+1}$ and $p_{n+1} = \hat{p}_{n+1}$. If we detect a collision as shown in Figure 4(c), we refer to $\hat{p}_{n+1}$ as penetration position and $\hat{v}_{n+1}$ as penetration velocity. The penetration direction is defined as $\hat{n} = (\hat{p}_{2,n+1} - \hat{p}_{1,n+1})/||\hat{p}_{2,n+1} - \hat{p}_{1,n+1}||_2$. Traditional simulation methods solves the post-collision velocities v_{n+1} using penetration velocity $\hat{v}_{n+1}$ and penetration direction $\hat{n}$, both of which may be discontinuous across optimization iterations due to time discretization. The TOI-Position proposed by Hu et al. (2020) computes TOI as the time spent after the penetration appears, i.e.,

$$TOI = d/\big((\hat{v}_{2,n+1} - \hat{v}_{1,n+1}) \cdot \hat{n}\big). \tag{3}$$

where $d = ||\hat{p}_{2,n+1} - \hat{p}_{1,n+1}||_2 - 2r$ is the penetration depth. In other words, the collision time is estimated by $(\Delta t - TOI)$ after the balls are in position p_n. Then the post-collision position is adjusted by $p_{n+1} = p_n + \hat{v}_{n+1} \cdot (\Delta t - TOI) + \tilde{v}_{n+1} \cdot TOI$, where $\tilde{v}_{n+1}$ is the velocity after resolving collision, which depends on elasticity. However, we argue that adjusting the position only is not enough since the post-collision velocity is also incorrectly estimated. To improve that, we consider the collision position $\bar{p}_{n+1}$ and velocity $\bar{v}_{n+1}$

$$\bar{v}_{n+1} = v_n + u_n(\Delta t - TOI), \tag{4}$$
$$\bar{p}_{n+1} = p_n + \hat{v}_{n+1}(\Delta t - TOI). \tag{5}$$

The collision direction based on the collision position are then

$$\bar{n} = (\bar{p}_{2,n+1} - \bar{p}_{1,n+1})/||\bar{p}_{2,n+1} - \bar{p}_{1,n+1}||_2. \tag{6}$$

Figure 4(c) shows a demonstration of collision position $\bar{p}_{n+1}$ and collision direction $\bar{n}$. We propose to use collision velocity $\bar{v}_{n+1}$ and collision direction $\bar{n}$ to solve for post-collision velocities v_{n+1}. In this way, the post-collision velocities would not suffer from the sudden jump as we observed in Figure 4(a) and 4(b). Post-collision position p_{n+1} can be determined accordingly.

5. Experiments

5.1. Optimal control formulation

The problem is formulated as an optimal control problem in continuous-time with state jumps:

$$\begin{aligned} \underset{u(\cdot)}{\text{minimize}} \quad & \phi(s(T)) + \int_0^T L(s(t), u(t))\mathrm{d}t, \\ \text{subject to} \quad & \dot{s}(t) = f(s(t), u(t)), t \in [0, T] \setminus \cup_{k\in\mathcal{K}}\{\gamma_k\}, \\ & \psi(s(\gamma_k^-)) = 0, \\ & s(\gamma_k^+) = g(s(\gamma_k^-)). \end{aligned} \tag{7}$$

Here we use $s = [p, v]$ to denote the positions and velocities of two balls as the state variable. f denotes the state dynamics under external forces u; $\cup_{k\in\mathcal{K}}\{\gamma_k\}$ denotes the set of collision instances which is characterized by the collision detection function ψ; and g denotes the effect of collisions on the state. We choose the terminal cost to be $\phi(s(T)) = ||p_2(T)||_2^2$ to capture our goal of Ball 2 reaching the origin and running cost to be $L(s, u) = \epsilon||u||_2^2$ to penalize large control inputs. In the experiments, we set $\epsilon = 0.01$. See the appendix of Hu et al. (2022) for the analytical solution of this optimal control problem.

To solve the above problem approximately, we discretize the problem into

$$\underset{u_0,\ldots,u_{N-1}}{\text{minimize}} \quad \phi(s_N) + \sum_{i=0}^{N-1} L(s_i, u_i)\Delta t, \tag{8}$$

$$\text{subject to} \quad s_{i+1} = \texttt{step}(s_i, u_i, \Delta t). \tag{9}$$

where the `step` function takes the current state and control as inputs and calculates the next time step state based on dynamics and collisions. In our experiments, the simulation time is $T = 1s$. The time period is discretized into $N = 480$ steps, i.e., $\Delta t = 1/480$. As the simulation can be made differentiable, we can differentiate through the `step` function and solve for the optimal control sequence directly using gradient descent.

5.2. A Single-Collision Example (Motivating Problem)

In this section, we demonstrate our proposed techniques using our motivating problem (Figure 1). We have two balls, of the same size (radius $r = 0.2$) on a plane. The collision between the two balls is frictionless and totally elastic. The initial positions of the balls are $p_{1,0} = [-1, -2]$ and $p_{2,0} = [-1, -1]$ and the initial velocities are $v_{1,0} = v_{2,0} = [0, 0]$. We can freely choose control inputs as forces acted on the Ball 1, i.e., $u_{1,n}$. The goal is to push Ball 1 to strike Ball 2 so that Ball 2 would be close to the origin at the end of the simulation. This goal can be formulated as an optimal control problem (8), where we set $\epsilon = 0.01$. We initiate our control sequence as a constant force $u_{1,n} = [0, 3], n = 0, ..., N - 1$.

5.2.1. PERFORMANCE OF EXISTING METHODS

We implemented several differentiable simulation methods discussed in Section 2.1 - linear complementarity problems (LCP), convex optimization problem (Convex), direct velocity impulse (Direct), compliant model (Compliant) and position-based dynamics (PBD).

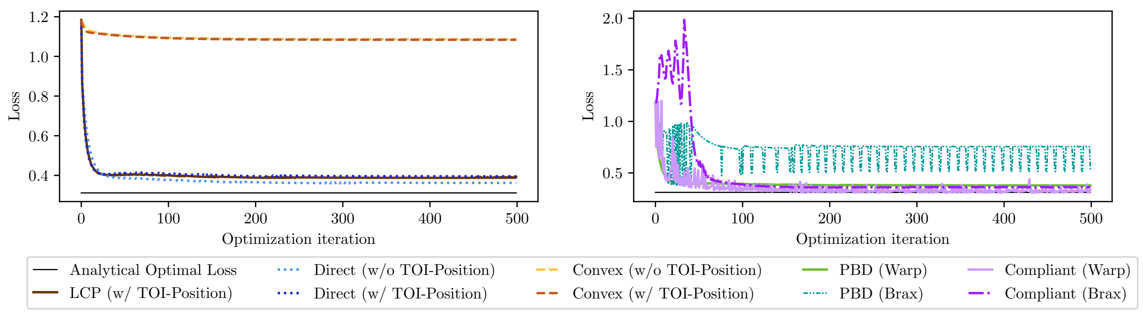

Figure 5: Single-collision: learning curves of different existing differentiable simulation methods. **Left**: Methods based on velocity impulses. **Right**: Compliant models and PBD.

Figure 5 shows the learning curves of existing methods. The left panel shows all the velocity-impulse-based methods and none of them converges to the analytical optimal loss. The right panel shows non-velocity-impulse-based methods, where many spikes exist in the learning process except PBD implemented in Warp.

The challenge encountered by existing methods can be further observed from the learned control shown in Figure 6. Even though in the x direction some of the learned control sequences match the shape of the analytical one, in the y direction none of the existing methods is able to learn the correct shape. Specifically, the pre-collision control sequence in the y direction should increase over time while all the learned pre-collision control sequences decrease over time.

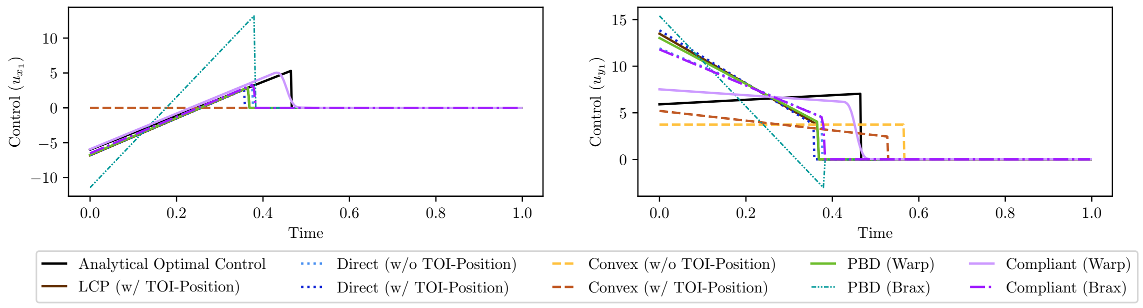

Figure 6: Single-collision: learned control from existing differentiable simulation methods.

5.2.2. PERFORMANCE OF TOI-VELOCITY

We implement TOI-Velocity using PyTorch and Taichi. The learning curves in Figure 7 show that both implementations are able to converge to the analytical optimal loss. Figure 8 shows the learned control sequences. Both implementations can learn the shape of analytical optimal control and the PyTorch implementation matches the analytical solution better. When comparing Figure 8 with the results of existing methods in Figure 6, we see a clear improvement especially in the y direction, since increasing pre-collision sequences are learned correctly.

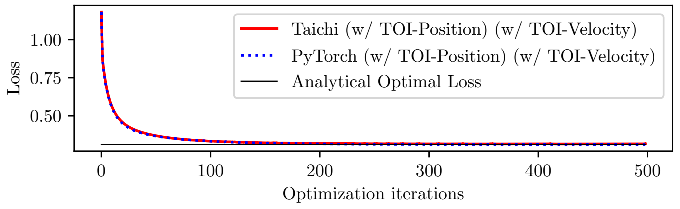

Figure 7: Single-collision: learning curves of our proposed method

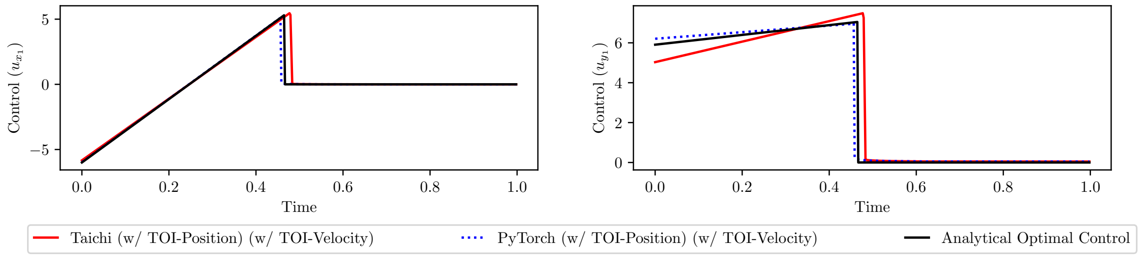

Figure 8: Single-collision: learned control sequence of our proposed method.

5.3. A Multiple-Collision Example

In this section, we study the system shown in Figure 9, where multiple collisions could happen. We have two balls, of the same size (radius $r = 0.2$) on a plane. The initial positions of the balls are $p_{1,0} = [0.25, -0.3]$ and $p_{2,0} = [-0.5, 0.6]$ and the initial velocities are $v_{1,0} = v_{2,0} = [0, 0]$. There is a wall at location $y = 1$. Both the ball-ball collision and ball-wall collision are frictionless and totally elastic. We can freely choose control inputs as forces acted on the first ball. The goal here is to push Ball 1 to strike Ball 2 so that Ball 2 would be close to the origin at the end of the simulation. The problem can be formulated as (7). Figure 9(a) shows the trajectory before optimization where a constant control of $u_{1,n} = [-3.5, 3.0], n = 0, ..., N-1$ is applied to Ball 1. This trajectory involves two ball-ball collisions and one ball-wall collision. Figure 9(b) shows the analytical optimal trajectory, which involves one ball-ball collision and one ball-wall collision.

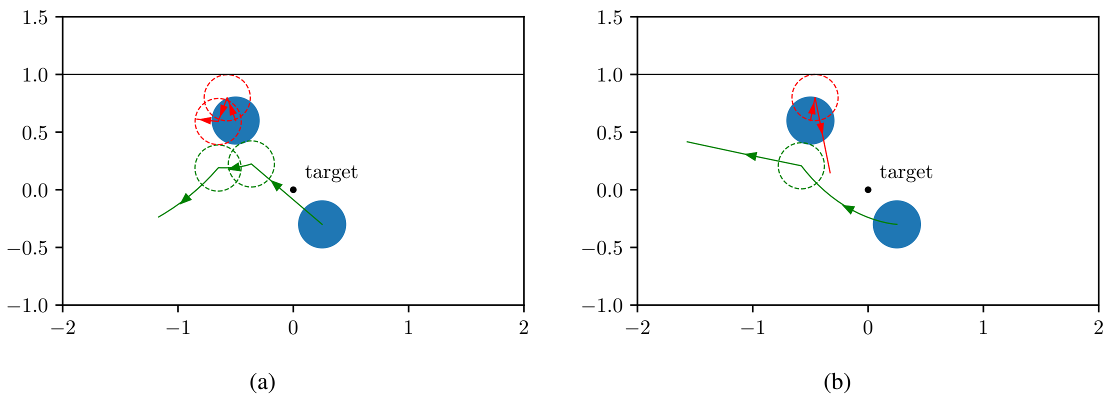

Figure 9: Multiple-collision: (a) Trajectory before optimization; (b) Analytical optimal trajectory.

5.3.1. PERFORMANCE OF EXISTING METHODS

Figure 10 shows the learning curves of existing differentiable simulation methods and none of them converges to the analytical optimal loss. The left panel shows results of velocity-impulse-based methods and for each method, there exists certain time steps where the loss increases over the iterations. This increase of loss is similar to the one observed in Figure 3, indicating wrong gradients calculated by the differentiable simulations. The right panel shows results of non-velocity-impulse-based methods. The learning using the Brax implementation of PBD is unstable as there are many spikes. The other learning curves are smooth but there is a clear gap between the analytical optimal loss and the converged values. Figure 11 shows the control sequences learned by existing differentiable simulation methods and none of them are close to the analytical optimal control.

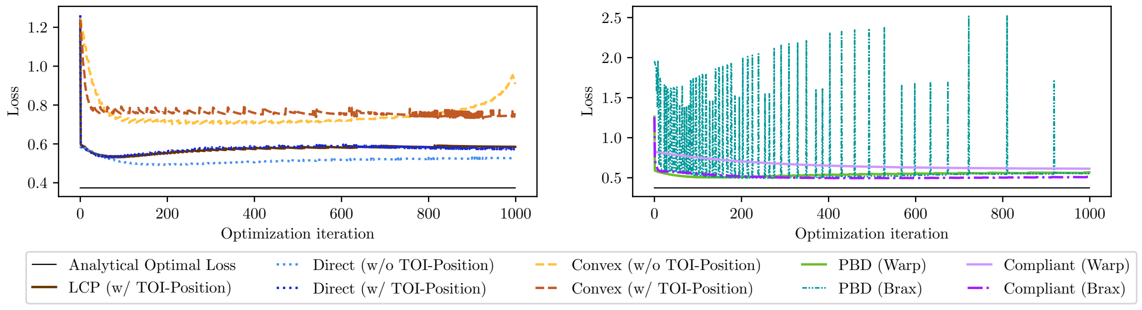

Figure 10: Multiple-collision: learning curves of different existing differentiable simulation methods. **Left**: Methods based on velocity impulses. **Right**: Compliant models and PBD.

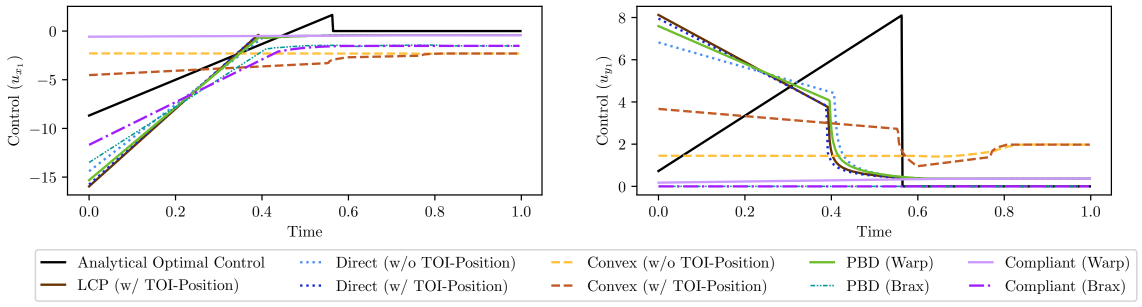

Figure 11: Multiple-collision: learned control sequences of existing methods.

5.3.2. PERFORMANCE OF TOI-VELOCITY

Applying TOI-Velocity can successfully solve the optimal control problem. Figure 12 shows that both Taichi and PyTorch implementation converges to the analytical optimal loss. Figure 13 shows that the learned control sequences match the analytical optimal control very well. By comparing these results with existing methods, we can conclude that adding the TOI-velocity improves the gradient calculation of velocity-impulse-based differentiable simulation methods. These improved gradients enable us to learn optimal control simply using gradient descent.

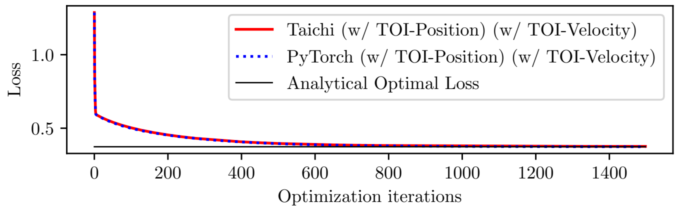

Figure 12: Multiple-collision: learning curves of our proposed method.

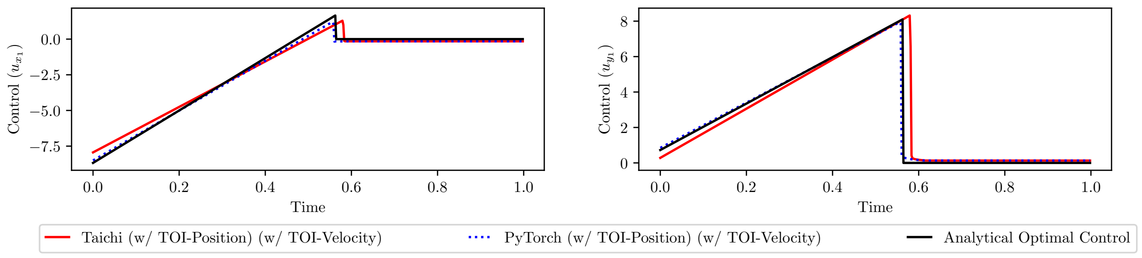

Figure 13: Multiple-collision: learned control of our proposed method.

5.4. Ablation Study

We have shown that adding TOI-Velocity enables us to successfully learn optimal control, but it is unclear whether TOI-Position is still necessary once TOI-Velocity is applied. Table 1 shows an ablation study on TOI-Position and TOI-Velocity for the two examples in Section 5.2 and 5.3. We find that only by applying both TOI-Position and TOI-Velocity can we end up with a loss value close to the analytical optimal loss in both examples. Without TOI-Position, the computed gradients would be wrong, which affects gradient-based learning.

Table 1: Ablation study on TOI-Position and TOI-Velocity

TOI-Position	TOI-Velocity	single-collision example	multiple-collision example
✗	✗	0.3616	0.5261
✓	✗	0.3949	0.5841
✗	✓	0.4797	0.6142
✓	✓	0.3151	0.3785
Analytical optimal loss		**0.3115**	**0.3737**

6. Conclusion

In this paper we propose a novel technique, TOI-Velocity, to reduce discontinuity caused by time discretization in physics simulation. Our proposed method is designed to improve gradient computation in differentiable simulation with contacts. We demonstrate TOI-Velocity in two optimal control examples. Our results show that applying TOI-Velocity together with TOI-Position is the only differentiable simulation implementation that can successfully learn the optimal control in these examples.

Acknowledgments

The codebase associated with this work will be released at https://github.com/DesmondZhong/diff_sim_improve_grads.

References

Akshay Agrawal, Brandon Amos, Shane Barratt, Stephen Boyd, Steven Diamond, and J Zico Kolter. Differentiable convex optimization layers. *Advances in neural information processing systems*, 32, 2019.

Justin Carpentier and Nicolas Mansard. Analytical derivatives of rigid body dynamics algorithms. In *Robotics: Science and systems (RSS 2018)*, 2018.

Ricky T. Q. Chen, Brandon Amos, and Maximilian Nickel. Learning neural event functions for ordinary differential equations. In *International Conference on Learning Representations*, 2021.

Filipe de Avila Belbute-Peres, Kevin Smith, Kelsey Allen, Josh Tenenbaum, and J Zico Kolter. End-to-end differentiable physics for learning and control. *Advances in neural information processing systems*, 31, 2018.

Jonas Degrave, Michiel Hermans, Joni Dambre, et al. A differentiable physics engine for deep learning in robotics. *Frontiers in neurorobotics*, page 6, 2019.

Tao Du, Kui Wu, Pingchuan Ma, Sebastien Wah, Andrew Spielberg, Daniela Rus, and Wojciech Matusik. Diffpd: Differentiable projective dynamics. *ACM Transactions on Graphics (TOG)*, 41 (2):1–21, 2021.

C. Daniel Freeman, Erik Frey, Anton Raichuk, Sertan Girgin, Igor Mordatch, and Olivier Bachem. Brax - a differentiable physics engine for large scale rigid body simulation, 2021. URL http://github.com/google/brax.

Moritz Geilinger, David Hahn, Jonas Zehnder, Moritz Bächer, Bernhard Thomaszewski, and Stelian Coros. Add: Analytically differentiable dynamics for multi-body systems with frictional contact. *ACM Transactions on Graphics (TOG)*, 39(6):1–15, 2020.

Eric Heiden, Miles Macklin, Yashraj Narang, Dieter Fox, Animesh Garg, and Fabio Ramos. Disect: A differentiable simulation engine for autonomous robotic cutting. *arXiv preprint arXiv:2105.12244*, 2021a.

Eric Heiden, David Millard, Erwin Coumans, Yizhou Sheng, and Gaurav S Sukhatme. Neuralsim: Augmenting differentiable simulators with neural networks. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9474–9481. IEEE, 2021b.

Taylor A Howell, Simon Le Cleac'h, J Zico Kolter, Mac Schwager, and Zachary Manchester. Dojo: A differentiable simulator for robotics. *arXiv preprint arXiv:2203.00806*, 2022.

Wei Hu, Jihao Long, Yaohua Zang, Weinan E, and Jiequn Han. Solving optimal control of rigid-body dynamics with collisions using the hybrid minimum principle. *arXiv preprint arXiv:2205.08622*, 2022.

Yuanming Hu, Luke Anderson, Tzu-Mao Li, Qi Sun, Nathan Carr, Jonathan Ragan-Kelley, and Fredo Durand. Difftaichi: Differentiable programming for physical simulation. In *International Conference on Learning Representations*, 2020.

Quentin Le Lidec, Igor Kalevatykh, Ivan Laptev, Cordelia Schmid, and Justin Carpentier. Differentiable simulation for physical system identification. *IEEE Robotics and Automation Letters*, 6 (2):3413–3420, 2021.

Minchen Li, Zachary Ferguson, Teseo Schneider, Timothy Langlois, Denis Zorin, Daniele Panozzo, Chenfanfu Jiang, and Danny M Kaufman. Incremental potential contact: Intersection-and inversion-free, large-deformation dynamics. *ACM transactions on graphics*, 2020.

Yifei Li, Tao Du, Kui Wu, Jie Xu, and Wojciech Matusik. Diffcloth: Differentiable cloth simulation with dry frictional contact. *arXiv preprint arXiv:2106.05306*, 2021.

Junbang Liang and Ming C. Lin. Differentiable Physics Simulation. In *ICLR 2020 Workshop on Integration of Deep Neural Models and Differential Equations*, 2020.

Junbang Liang, Ming Lin, and Vladlen Koltun. Differentiable cloth simulation for inverse problems. *Advances in Neural Information Processing Systems*, 32, 2019.

Quentin Le Lidec, Louis Montaut, Cordelia Schmid, Ivan Laptev, and Justin Carpentier. Augmenting differentiable physics with randomized smoothing. *arXiv preprint arXiv:2206.11884*, 2022.

M Macklin, K Erleben, M Müller, N Chentanez, S Jeschke, and TY Kim. Primal/dual descent methods for dynamics. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 1–12, 2020.

Miles Macklin. Warp: A high-performance python framework for gpu simulation and graphics. `https://github.com/nvidia/warp`, March 2022. NVIDIA GPU Technology Conference (GTC).

Miles Macklin, Matthias Müller, and Nuttapong Chentanez. Xpbd: position-based simulation of compliant constrained dynamics. In *Proceedings of the 9th International Conference on Motion in Games*, pages 49–54, 2016.

Matthias Müller, Bruno Heidelberger, Marcus Hennix, and John Ratcliff. Position based dynamics. *Journal of Visual Communication and Image Representation*, 18(2):109–118, 2007.

J. Krishna Murthy, Miles Macklin, Florian Golemo, Vikram Voleti, Linda Petrini, Martin Weiss, Breandan Considine, Jérôme Parent-Lévesque, Kevin Xie, Kenny Erleben, Liam Paull, Florian Shkurti, Derek Nowrouzezahrai, and Sanja Fidler. gradsim: Differentiable simulation for system identification and visuomotor control. In *International Conference on Learning Representations*, 2021.

Yi-Ling Qiao, Junbang Liang, Vladlen Koltun, and Ming Lin. Scalable differentiable physics for learning and control. In *International Conference on Machine Learning*, pages 7847–7856. PMLR, 2020.

Yi-Ling Qiao, Junbang Liang, Vladlen Koltun, and Ming C Lin. Efficient differentiable simulation of articulated bodies. In *International Conference on Machine Learning*, pages 8661–8671. PMLR, 2021.

Changkyu Song and Abdeslam Boularias. Identifying mechanical models of unknown objects with differentiable physics simulations. In *Proceedings of the 2nd Conference on Learning for Dynamics and Control*, volume 120 of *Proceedings of Machine Learning Research*, pages 749–760. PMLR, 2020a.

Changkyu Song and Abdeslam Boularias. Learning to slide unknown objects with differentiable physics simulations. In *Robotics science and systems*, 2020b.

Michael Strecke and Joerg Stueckler. Diffsdfsim: Differentiable rigid-body dynamics with implicit shapes. In *2021 International Conference on 3D Vision (3DV)*, pages 96–105. IEEE, 2021.

Hyung Ju Terry Suh, Tao Pang, and Russ Tedrake. Bundled gradients through contact via randomized smoothing. *IEEE Robotics and Automation Letters*, 7(2):4000–4007, 2022a.

Hyung Ju Terry Suh, Max Simchowitz, Kaiqing Zhang, and Russ Tedrake. Do differentiable simulators give better policy gradients? In *International Conference on Machine Learning*, pages 20668–20696. PMLR, 2022b.

Emanuel Todorov. A convex, smooth and invertible contact model for trajectory optimization. In *2011 IEEE International Conference on Robotics and Automation*, pages 1071–1076, 2011.

Emanuel Todorov. Convex and analytically-invertible dynamics with contacts and constraints: Theory and implementation in MuJoCo. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6054–6061, 2014.

Emanuel Todorov, Tom Erez, and Yuval 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.

Dylan Turpin, Liquan Wang, Eric Heiden, Yun-Chun Chen, Miles Macklin, Stavros Tsogkas, Sven Dickinson, and Animesh Garg. Grasp'd: Differentiable contact-rich grasp synthesis for multi-fingered hands. In *European Conference on Computer Vision*, pages 201–221. Springer, 2022.

Keenon Werling, Dalton Omens, Jeongseok Lee, Ioannis Exarchos, and C Karen Liu. Fast and feature-complete differentiable physics for articulated rigid bodies with contact. *arXiv preprint arXiv:2103.16021*, 2021.

Jie Xu, Tao Chen, Lara Zlokapa, Michael Foshey, Wojciech Matusik, Shinjiro Sueda, and Pulkit Agrawal. An End-to-End Differentiable Framework for Contact-Aware Robot Design. In *Proceedings of Robotics: Science and Systems*, Virtual, July 2021.

Jie Xu, Miles Macklin, Viktor Makoviychuk, Yashraj Narang, Animesh Garg, Fabio Ramos, and Wojciech Matusik. Accelerated policy learning with parallel differentiable simulation. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=ZSKRQMvttc.

Shuqi Yang, Xingzhe He, and Bo Zhu. Learning physical constraints with neural projections. *Advances in Neural Information Processing Systems*, 33:5178–5189, 2020.

Yaofeng Desmond Zhong, Biswadip Dey, and Amit Chakraborty. Extending Lagrangian and Hamiltonian neural networks with differentiable contact models. *Advances in Neural Information Processing Systems*, 34, 2021.

Yaofeng Desmond Zhong, Jiequn Han, and Georgia Olympia Brikis. Differentiable physics simulations with contacts: Do they have correct gradients wrt position, velocity and control? *arXiv preprint arXiv:2207.05060*, 2022.