DIFFERENTIABLE PHYSICS SIMULATION

Junbang Liang **Ming Lin**
University of Maryland, College Park

ABSTRACT

Differentiable physics simulation is a powerful family of new techniques that applies gradient-based methods to learning and control of physical systems. It enables optimization for control, and can also be integrated into neural network frameworks for performing complex tasks. We believe that differentiable physics simulation should be a key component for neural networks to bridge the gap between training performance and the generality to previously unseen real-world inputs. However, realizing a practical differentiable simulation is still challenging because of its high dimensionality and fragmented computation flow. In this paper, we motivate the importance of differentiable physics simulation, describe its current challenges, introduce state-of-the-art approaches, and discuss potential improvements and future directions.

1 INTRODUCTION

Recent advances in deep learning have shown its promising ability to provide statistical approximations of many problems from data. However, there is still a gap between accuracy during training and during application and inference, simply because the entire set of inputs cannot be enumerated, resulting to the difference between distributions of training data and test data. The unknown or incorrect extrapolating behaviors of neural networks can sometimes be potentially catastrophic, when they are applied to control of dynamical systems, such as autonomous driving or robot arms. Many previous works have been studying on improving the generality using various techniques, the most popular one being domain transfer (Taylor & Stone, 2007).

Learning from simulation, as an alternative, has become increasingly more important and useful in many learning tasks. It eases the burden of collecting and labeling numerous diverse data from the real world, enabling problems with more complex definition to be modeled and learned. More importantly, validated and calibrated simulation provides data that is close to ground-truth for neural networks, and can thereby let the network learn from both successful and failure cases to obtain a more unbiased model.

Previous usages of simulation are mostly on data generation (Liang & Lin, 2019; Yang et al., 2017) or reinforcement learning (Degris et al., 2012), due to the fact that most of current simulation algorithms can only provide forward results but not backpropagated gradients of the input. We observe that many learning-based tasks involving simulation can be greatly improved if gradients of simulation can be provided. First, reinforcement learning tasks can benefit from gradients of the corresponding simulation by directly optimize the target reward, which can improve the convergence speed because it turns the reinforcement learning problem into a better conditioned optimization or planning problem. Next, other learning tasks that use simulation as training data can be extended to train on unlabeled real-world data. Given a set of output estimated parameters, differentiable simulation can be embedded as the last part of the network to provide simulated results so that it can be compared directly with the input. Last, given a differentiable simulation as a function, we can even approximate the simulation as a neural network in order to provide a faster approximated results in some speed-demand applications, a pioneer work being neural-ODE (Chen et al., 2018).

Thanks to the publicly available auto-differentiation tools, such as PyTorch (Steiner et al., 2019), computing gradients in such a complex simulation system becomes much easier. However, challenges still exist for realizing a general and fast differentiable simulation. First, the simulation problem has often a high dimensionality of input variables with irregular interaction patterns. Although a rigid body has only 6 degree of freedom, a deformable body can usually have tens of thousands

of vertices describing its current shape, each being a variable to solve. Next, the simulation process is often fragmented and hard to be vectorized. Each vertex often interacts with its neighboring vertices geodesically (as internal forces) and geometrically (as repulsive contact forces), being applied to nonlinear physical models and solved for the implicit position update. The forces are usually computed case-by-base based on the current material and geometry of the vertex, not to mention consistently changing contact status between objects that affects the force computation. These are all critical parts to be considered when we are trying to design a more neural-network-friendly simulation.

In the following sections, we first discuss possible simplifications and design rationales toward a generally useful differentiable physics engine. Then, we introduce current state-of-the-art that has addressed the above challenges to some extent, and show some promising results from the current work. Lastly we conclude the current progress point towards future improvements.

2 Towards General Differentiable Physics

General differentiable physics simulation for all phenomena is difficult. Different types of materials often need different physical models due to its dominant physical behaviors. For example, rigid body is the simplest because it has only one global transformation as its status. On the contrary, a deformable body will need to come with its rest shape and compute the internal forces based on how it is stretched or compressed, as compared to the rest pose. Fluid, as one of the most difficult materials to simulate, has to obey different rules of density, pressure and velocity, according to its compressibility. For real-world applications, we posit that a differentiable simulation modeling both rigid bodies and deformable bodies is adequate for most cases, because the physical interactions between these two materials are most in need among common machine-learning tasks such as robot control, while other types of materials have very little impact on these tasks.

Based on the challenges as mentioned before, we believe that a good differentiable simulation algorithm should suffice three rationales below:

1. **Vectorization friendly.** A computation graph is needed in order to compute the gradients in the backpropagation process of the simulation. Fragmented operations during the forward pass will unnecessarily enlarge the graph, slowing down the computation speed and taking more memory. A properly vertorized computation flow will boost up the simulation speed.
2. **GPU friendly.** Since most neural networks are trained in GPUs, it is desired to also have the differentiable simulation applicable in GPU as well. When some part of the simulation algorithm cannot be represented as tensor operations, a GPU version will be needed to avoid data transfer to CPUs and possible performance bottleneck.
3. **Supports sparse operations.** Since many force computations are related to the neighboring status of a vertex, an algorithm will be needed to support fast queries and operations that are sparse compared to the problem space. It is hard to achieve since it is often contradictory to the previous two rationales.

3 Current Progress

With recent advances in deep learning, there has been increasing interest in differentiable physics simulation, which can be combined with other learning methods to provide physically consistent predictions. de Avila Belbute-Peres et al. (2018) and Degrave et al. (2019) proposed rigid body simulators using a static formulation of the linear complementarity problem (LCP) (Cottle, 2009). Hu et al. (2019) implemented a differentiable simulator for soft robots based on the Material Point Method (MPM). They store the object data at every simulation step so that the gradient can be computed out of the box. The state-of-the-art work is from Liang et al. (2019), where they proposed a method to differentiate cloth simulation. It is the first work to tackle high dimensional simulation problem and to propose a general differentiable collision handling algorithm.

In general, they follow the computation flow of the common approach to cloth simulation: discretization using the finite element method, integration using implicit Euler, and collision response on impact zones. They use implicit differentiation in the linear solve and the optimization, in order

to compute the gradient with respect to the input parameters. The discontinuity introduced by the collision response is negligible because the discontinuous states constitute a zero-measure set. During the backpropagation in the optimization, the gradient values can be directly computed after QR decomposition of the constraint matrix. In their pipeline, there are several techniques that can be employed in other differentiable simulations.

3.1 Derivatives of the Physics Solve

In modern simulation algorithms, implicit Euler is often used for stable integration results. Thus the mass matrix $\mathbf{M}$ often includes the Jacobian of the forces. It is denoted as $\hat{\mathbf{M}}$ in order to mark the difference. A linear solve will be needed to compute the acceleration since it is time-consuming to compute $\hat{\mathbf{M}}^{-1}$. They use implicit differentiation to compute the gradients of the linear solve. Given an equation $\hat{\mathbf{M}}\mathbf{a} = \mathbf{f}$ with a solution $\mathbf{z}$ and the propagated gradient $\frac{\partial \mathcal{L}}{\partial \mathbf{a}}|_{\mathbf{a}=\mathbf{z}}$, where $\mathcal{L}$ is the task-specific loss function, they derived the implicit differentiation form

$$\hat{\mathbf{M}}\partial\mathbf{a} = \partial\mathbf{f} - \partial\hat{\mathbf{M}}\mathbf{a} \tag{1}$$

to derive the gradient as

$$\frac{\partial \mathcal{L}}{\partial \hat{\mathbf{M}}} = -\mathbf{d}_{\mathbf{a}}\mathbf{z}^{\top} \quad \frac{\partial \mathcal{L}}{\partial \mathbf{f}} = \mathbf{d}_{\mathbf{a}}^{\top}, \tag{2}$$

where $\mathbf{d}_{\mathbf{a}}$ is obtained from the linear system

$$\hat{\mathbf{M}}^{\top}\mathbf{d}_{\mathbf{a}} = \frac{\partial \mathcal{L}}{\partial \mathbf{a}}^{\top}. \tag{3}$$

3.2 Derivatives of the Collision Response

A general approach to integrating collision constraints into physics simulation has been proposed by de Avila Belbute-Peres et al. (2018). However, constructing a static LCP is often impractical in cloth simulation due to high dimensionality. Collisions and contacts happen at each step are very sparse compared to the complete set. Therefore, they use a dynamic approach that incorporates collision detection and response.

Collision handling in their implementation is based on impact zone optimization. It finds all colliding instances using continuous collision detection and sets up the constraints for all collisions. In order to introduce minimum change to the original mesh state, a QP problem is developed to solve for the constraints. Since the signed distance function is linear in $\mathbf{x}$, the optimization takes a quadratic form:

$$\underset{\mathbf{z}}{\text{minimize}} \quad \frac{1}{2}(\mathbf{z}-\mathbf{x})^{\top}\mathbf{W}(\mathbf{z}-\mathbf{x}) \tag{4}$$

$$\text{subject to} \quad \mathbf{G}\mathbf{z} + \mathbf{h} \leq \mathbf{0} \tag{5}$$

where $\mathbf{W}$ is a constant diagonal weight matrix related to the mass of each vertex, and $\mathbf{G}$ and $\mathbf{h}$ are constraint parameters. They further denote the number of variables and constraints by n and m, *i.e.* $\mathbf{x} \in \mathbb{R}^n$, $\mathbf{h} \in \mathbb{R}^m$, and $\mathbf{G} \in \mathbb{R}^{m \times n}$. Note that this optimization is a function with inputs $\mathbf{x}$, $\mathbf{G}$, and $\mathbf{h}$, and output $\mathbf{z}$. The goal here is to derive $\frac{\partial \mathcal{L}}{\partial \mathbf{x}}$, $\frac{\partial \mathcal{L}}{\partial \mathbf{G}}$, and $\frac{\partial \mathcal{L}}{\partial \mathbf{h}}$ given $\frac{\partial \mathcal{L}}{\partial \mathbf{z}}$, where $\mathcal{L}$ refers to the loss function.

When computing the gradient using implicit differentiation (Amos & Kolter, 2017), the dimensionality of the linear system can be very high. Their key observation here is that $n \gg m > \text{rank}(\mathbf{G})$, since one contact often involves 4 vertices (thus 12 variables) and some contacts may be linearly dependent (*e.g.* multiple adjacent collision pairs). They minimize the size of the linear equation based on the QR decomposition of $\mathbf{G}$, which is the key to accelerate backpropagation of high dimensional QP problems.

4 Experimental Results

In this section we show some experiment results from state-of-the-art (Liang et al., 2019). These experiments demonstrate the great potential of differentiable simulation being used in learning tasks such as hidden parameter estimation and control.

4.1 MATERIAL ESTIMATION

In this experiment, the aim is to learn the material parameters of cloth from observation. The scene features a piece of cloth hanging under gravity and subjected to a constant wind force. The material model consists of three parts: density d, stretching stiffness $\mathbf{S}$, and bending stiffness $\mathbf{B}$. The stretching stiffness quantifies how large the reaction force will be when the cloth is stretched out; the bending stiffness models how easily the cloth can be bent and folded.

Tab. 1 shows the estimation result. They achieve a much smaller error in most measurements in comparison to the baselines. The table shows that the linear part of the stiffness matrix is optimized well. With the computed gradient using our model, one can effectively optimize the unknown parameters that dominate the cloth movement to fit the observed data.

Method	Runtime (sec/step/iter)	Density error (%)	Linear streching stiffness error (%)	Bending stiffness error (%)	Simulation error (%)
Baseline	-	68 ± 46	160 ± 119	$\mathbf{70 \pm 42}$	12 ± 3.0
L-BFGS	2.89 ± 0.02	4.2 ± 5.6	72 ± 90	$\mathbf{70 \pm 43}$	4.9 ± 3.3
Liang et al. (2019)	$\mathbf{2.03 \pm 0.06}$	$\mathbf{1.8 \pm 2.0}$	$\mathbf{45 \pm 41}$	77 ± 36	$\mathbf{1.6 \pm 1.4}$

Table 1: Results on the material parameter estimation task. Lower is better. Values of the material parameters are the Frobenius norms of the difference normalized by the Frobenius norm of the target. Values of the simulated result are the average pairwise vertex distance normalized by the size of the cloth. The gradient-based method yields much smaller errors than the baselines.

4.2 MOTION CONTROL

They further demonstrate the power of the differentiable simulator by optimizing control parameters. The task is to drop a piece of cloth into a basket. The cloth is originally placed on a table that is away from the basket. The system then applies external forces to the corners of the cloth to lift it and drop it into the basket. The external force is applied for 3 seconds and can be changed during this period. The basket is a box with an open top. A planar obstacle is placed between the cloth and the basket to increase the difficulty of the task.

Tab. 2 shows the performance of the different methods and their sample complexity. The error shown in the table is the distance defined above normalized by the size of the cloth. Their method achieves the best performance with a much smaller number of simulation steps. The bottom of the basket in the setting has the same size as the cloth, so a normalized error of less than 50%, as they achieved, implies that the cloth is successfully dropped into the basket.

Method	Error (%)	Samples
Point mass	111	–
PPO	432	10,000
Liang et al. (2019)	**17**	**53**

Table 2: Motion control results. The table reports the smallest distance to the target position, normalized by the size of the cloth, and the number of samples used during training.

5 CONCLUSION AND DISCUSSION

We stated the importance of differentiable simulation, pointed out key features and design rationales for the general implementation, and demonstrated its potential by introducing the algorithms and experiments from state-of-the-art. By making use of the gradients from the physically-aware simulation, differentiable simulation can optimize the unknown parameters faster and more accurately than gradient-free methods. Using differentiable simulation, we can learn the intrinsic properties of objects from observation.

The limitation of the current work is that the current simulation architecture is not optimized for large-scale vectorized operations, which introduces some overhead. This can be addressed by a specialized, optimized simulation system based solely on tensor operations. Also, there are still challenges to be addressed, when simulating rigid bodies and deformable bodies together, one of which is the differentiable contact force coupling between different materials.

REFERENCES

Brandon Amos and J. Zico Kolter. OptNet: Differentiable optimization as a layer in neural networks. In *International Conference on Machine Learning (ICML)*, 2017.

Tian Qi Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. In *Advances in neural information processing systems*, pp. 6571–6583, 2018.

Richard W Cottle. *Linear Complementarity Problem*. Springer, 2009.

Filipe de Avila Belbute-Peres, Kevin A. Smith, Kelsey Allen, Josh Tenenbaum, and J. Zico Kolter. End-to-end differentiable physics for learning and control. In *Advances in Neural Information Processing Systems*, 2018.

Jonas Degrave, Michiel Hermans, Joni Dambre, and Francis wyffels. A differentiable physics engine for deep learning in robotics. *Frontiers in Neurorobotics*, 13, 2019.

Thomas Degris, Patrick M Pilarski, and Richard S Sutton. Model-free reinforcement learning with continuous action in practice. In *2012 American Control Conference (ACC)*, pp. 2177–2182. IEEE, 2012.

Yuanming Hu, Jiancheng Liu, Andrew Spielberg, Joshua B. Tenenbaum, William T. Freeman, Jiajun Wu, Daniela Rus, and Wojciech Matusik. ChainQueen: A real-time differentiable physical simulator for soft robotics. In *International Conference on Robotics and Automation (ICRA)*, 2019.

Junbang Liang and Ming C Lin. Shape-aware human pose and shape reconstruction using multi-view images. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 4352–4362, 2019.

Junbang Liang, Ming Lin, and Vladlen Koltun. Differentiable cloth simulation for inverse problems. In *Advances in Neural Information Processing Systems*, pp. 771–780, 2019.

Benoit Steiner, Zachary DeVito, Soumith Chintala, Sam Gross, Adam Paszke, Francisco Massa, Adam Lerer, Gregory Chanan, Zeming Lin, Edward Yang, et al. PyTorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, 2019.

Matthew E Taylor and Peter Stone. Cross-domain transfer for reinforcement learning. In *Proceedings of the 24th international conference on Machine learning*, pp. 879–886. ACM, 2007.

Shan Yang, Junbang Liang, and Ming C Lin. Learning-based cloth material recovery from video. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 4383–4393, 2017.