Rethinking Optimization with Differentiable Simulation from a Global Perspective

Anonymous Author(s)
Affiliation
Address
email

Abstract: Differentiable simulation is a promising toolkit for fast gradient-based policy optimization and system identification. However, existing approaches to differentiable simulation have largely tackled scenarios where obtaining smooth gradients has been relatively easy, such as systems with mostly smooth dynamics. In this work, we study the challenges that differentiable simulation presents when it is not feasible to expect that a single descent reaches a global optimum, which is often a problem in contact-rich scenarios. We analyze the optimization landscapes of diverse scenarios that contain both rigid bodies and deformable objects. In dynamic environments with highly deformable objects and fluids, differentiable simulators produce rugged landscapes with nonetheless useful gradients in some parts of the space. We propose a method that combines Bayesian optimization with semi-local 'leaps' to obtain a global search method that can use gradients effectively, while also maintaining robust performance in regions with noisy gradients. We show that our approach outperforms several gradient-based and gradient-free baselines on an extensive set of experiments in simulation, and also validate the method using experiments with a real robot and deformables.

Keywords: Differentiable simulation, Global optimization, Deformable Objects

1 Introduction

Physics simulation is indispensable for robot learning: it has been widely used to generate synthetic training data, explore learning of complex sensorimotor policies, and also help anticipate the performance of various learning methods before deploying on real robots. An increasing volume of recent work attempts to *invert* physics engines: given simulation outputs (e.g. trajectories), infer the input parameters (e.g. physical properties of the scene, robot controls) that best explain the outputs [1, 2, 3, 4, 5]. Differentiable physics engines aim to offer a direct and efficient way to invert simulation, and could enable fast gradient-based policy optimization and system identification. Works that we survey in the background section show a number of recent successes. However, to highlight the most promising prospects of differentiable simulation, works in this space focus primarily on scenarios where obtaining smooth gradients is relatively easy.

With more research efforts gearing to develop (and use) differentiable physics engines, it is now crucial that we analyze the limits of these systems thoroughly. Surprisingly, efforts to investigate the *differentiability* of these simulators are far and few. One prior work [6] has highlighted a few fundamental limitations of differentiable simulation in the presence of rigid contacts, in low-dimensional systems. In this work, we first investigate the *quality* of gradients by visualizing loss landscapes through differentiable simulators for several robotic manipulation tasks of interest. For this analysis we create several new challenging environments, and also use environments from prior work.

Our main focus is to understand whether the limitations of rigid contacts are also prevalent across deformable object simulation. We show that in many cases obtaining well-behaved gradients is fea-

Submitted to the 6th Conference on Robot Learning (CoRL 2022). Do not distribute.

sible, but challenge the assumption that differentiable simulators provide easy landscapes in sufficiently interesting scenarios. We analyze scenarios with flexible deformable objects (cloth), plastics (clay), and fluids. Our visualizations of optimization landscapes uncover numerous local optima and plateaus, showing the need for extending gradient-based methods with global search. We propose a method that combines global search using Bayesian optimization with a semi-local search. Our semi-local strategy allows to make progress on parts of the landscape that are intractable for gradient descent alone. In our experiments we show visual, quantitative and qualitative analysis of the optimization problems arising from simulations with deformables in a variety of scenarios. We also validate our proposed approach on a real robot interacting with cloth, where our aim is to identify the properties of the cloth that make the motion of the simulated cloth match the real one.

2 Background

Differentiable Simulation: The earliest differentiable simulators [7, 8] focused solely on rigid-body dynamics, often operating only over a small number of predefined object primitives. A number of approaches subsequently focused on arbitrary rigid body shapes [9, 10], articulations [11, 12, 13], accurate contacts [14, 15, 16], scalability [17], speed [18, 19], and multiphysics [1]. Differentiable simulation has also been explored in the context of deformable objects [20], cloth [21], fluids [22], robotic cutting [23] and other scientific and engineering phenomena [24, 25, 26]. While several approaches sought to achieve physically accurate forward simulations and their gradients, they did not rigorously study the impact of their loss landscapes on inference. Most modern simulators are plagued by two fundamental issues. First, they rely on gradient descent as an inference mechanism, which makes the optimization dependent on a *good* initial guess. Second, the loss landscape for contact-rich scenarios is laden with discontinuities and spurious local optima. This is evident even in the most basic case of a single bouncing ball as shown in [19]. In this work, we also verify that the issue is present in the more recent frameworks, such as Nimble [14] and Warp [27].

Global Search Methods: Global optimization tackles the problem of finding the global optimum in a given search space. In our case, this could constitute finding the optimal parameters for a controller (e.g. target position, velocity, force, torque) or physical parameters of a simulator to make the behavior of simulated objects match reality. Global optimization includes several broad families of methods: 1) space covering methods that systematically visit all parts of the space; 2) clustering methods that use cluster analysis to decide which areas of the space could be promising; 3) evolutionary methods that start from a broad set of candidates and evolve them towards exploring the promising regions; 4) Bayesian optimization (BO) that uses non-parametric approaches to keep track of a global model of the cost function and its uncertainty on the whole search space. The survey in [28] gives further details. Random search is also considered a global search method, and is one of the few methods that guarantees eventually finding a global optimum. While data efficiency can be a challenge, it is often a surprisingly robust baseline. Space covering and clustering methods face significant challenges when trying to scale to high dimensions. The recent focus in global optimization has been on evolutionary methods and BO, which can be successful on search spaces with thousands of dimensions. Hence, in this work we compare methods based on random search, evolutionary approaches and BO.

CMA-ES: One of the most versatile evolutionary methods is Covariance Matrix Adaptation - Evolution Strategies (CMA-ES) [29]. It samples a randomized 'generation' of points at each iteration from a multivariate Gaussian distribution. To evolve this distribution, it computes the new mean from a subset of points with the lowest cost from the previous generation. It also uses these best-performing points to update the covariance matrix. The next generation of points is then sampled from the distribution with the updated mean and covariance. CMA-ES succeeds in a wide variety of applications [30], and has competitive performance on global optimization benchmarks [31]. This method does not make any restrictive assumptions about the search space and can be used 'as-is' i.e. without tuning. However, this method is technically not fully global — while it can overcome shallow local optima, it can get stuck in deeper local optima.

Bayesian optimization (BO): BO views the problem of global search as seeking a point $\boldsymbol{x}^*$ to minimize a given cost function $f(\boldsymbol{x})$: $f(\boldsymbol{x}^*) = \min_{\boldsymbol{x}} f(\boldsymbol{x})$. At each trial, BO optimizes an auxiliary acquisition function to select the next promising $\boldsymbol{x}$ to evaluate. f is frequently modeled with a Gaussian process (GP): $f(\boldsymbol{x}) \sim \mathcal{GP}(m(\boldsymbol{x}), k(\boldsymbol{x}_i, \boldsymbol{x}_j))$. Modeling f with a GP allows to compute the posterior mean $\bar{f}(\boldsymbol{x})$ and uncertainty (variance) $Var[f(\boldsymbol{x})]$ for each candidate point $\boldsymbol{x}$. Hence, the acquisition function can select points to balance a high mean (exploitation) with high uncertainty (exploration). The kernel function $k(\cdot,\cdot)$ encodes similarity between inputs. If $k(\boldsymbol{x}_i, \boldsymbol{x}_j)$ is large for inputs $\boldsymbol{x}_i, \boldsymbol{x}_j$, then $f(\boldsymbol{x}_i)$ strongly influences $f(\boldsymbol{x}_j)$. One of the most widely used kernel functions is the Squared Exponential (SE): $k_{SE}(\boldsymbol{r} \equiv |\boldsymbol{x}_i - \boldsymbol{x}_j|) = \sigma_k^2 \exp\left(-\frac{1}{2}\boldsymbol{r}^T \operatorname{diag}(\boldsymbol{\ell})^{-2}\boldsymbol{r}\right)$, where σ_k^2, $\boldsymbol{\ell}$ are signal variance and a vector of length scales respectively. σ_k^2, $\boldsymbol{\ell}$ are hyperparameters and are optimized automatically by maximizing the marginal data likelihood. See [32] for further details.

3 BO-Leap: A Method for Global Search on Rugged Landscapes

We propose an approach for global search that can benefit from gradient-based descents and employs a semi-local strategy to make progress on rough optimization landscapes. To explore the loss landscape globally, we use Bayesian optimization (BO). BO models uncertainty over the loss and reduces uncertainty globally by exploring unseen regions. It also ensures to return to low-loss regions to further improve within the promising areas of the search space. BO uses an acquisition function to compute the most promising candidate to evaluate next. We treat each candidate as a starting point for a semi-local search. As we show in our experiments, the straightforward strategy of using gradient descent from each of these starting points does not ensure strong performance in scenarios with noisy gradients. Hence, we propose a hybrid descent strategy that combines gradient-free search with gradient-based descents. For this, we collect a small population of local samples and compute a sampling distribution based on CMA-ES. Instead of directly using the resulting mean and covariance to sample the next population (as CMA-ES would), we use gradient descent to evolve the distribution mean, then use this updated mean when sampling the next population.

We outline our method, BO with semi-local leaps (BO-Leap), in Algorithm 1. We also visualize the algorithm in Figure 1. We start by initializing a global model for the loss with a Gaussian process (GP) prior. Using the BO acquisition function, we sample a vector $\boldsymbol{x}_1$ of simulation or control parameters to evaluate. We then run a semi-local search from this starting point. For this, we initialize the population distribution $\mathcal{N}(\boldsymbol{\mu}_1 = \boldsymbol{x}_1, \sigma_1^2 \boldsymbol{C}_1)$ and sample K local candidates. Next, we update the distribution in a gradient-free way similar to the CMA-ES strategy. We then start a gradient descent from the updated mean $\boldsymbol{s}_1$. The gradient descent runs for at most J steps and is halted if the loss stagnates or increases for more than three steps. The gradient is clipped to avoid leaving the search space boundaries. When the semi-local search reaches a given number of steps (e.g. 100 in our experiments), we update the BO posterior and let BO pick a new starting point $\boldsymbol{x}_2$ globally. We add all the points that the semi-local search encounters to the set $\boldsymbol{S}_n$ that is used to compute the posterior for each BO trial.

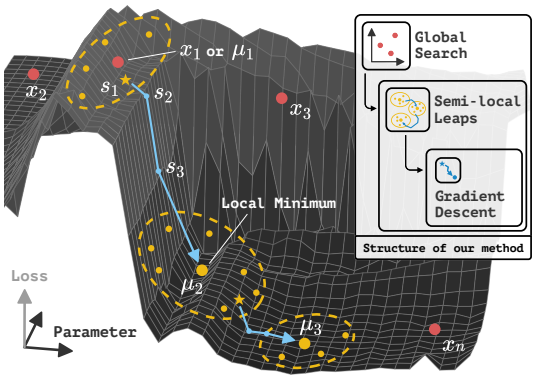

Figure 1: A conceptual illustration of BO-Leap.

Obtaining a noticeable improvement by incorporating gradients into a strategy based on CMA-ES is not trivial. One hybrid approach that seems conceptually sound proposes to shift the mean of each CMA-ES population by taking a step in the direction of the gradient [33]. In our preliminary experiments, taking a single gradient step was insufficient to significantly improve the performance of CMA-ES: the method from [33] performed worse than gradient-free CMA-ES. In contrast, our semi-local search strategy allows gradient-based descents to take large leaps on the parts of the landscape where gradients are relatively smooth. To prevent being mislead by unstable gradients and

Algorithm 1: BO-Leap : Bayesian Optimization with Semi-local Leaps

Initialize : $\boldsymbol{S}_0 \leftarrow \{\}$ ▷ Set of points seen so far (initially empty)
$\mathcal{L} \leftarrow \mathcal{GP}\big(m(\boldsymbol{x})=\boldsymbol{0},\, k(\boldsymbol{x},\boldsymbol{x'})\,|\boldsymbol{S}_0\big)$ ▷ Gaussian process as a global loss model
BO acquisition function ← LCB ▷ Lower Confidence Bound
for $n = 1..max_steps$ **do**
  $\mathcal{GP}_{\boldsymbol{S}_n}=\mathcal{GP}\big(m(\cdot),k(\cdot,\cdot)|\boldsymbol{S}_n\big)$ ▷ Compute GP posterior using Eq. 2.25-26 from [34]
  $\boldsymbol{x}_n \leftarrow \arg\min_{\boldsymbol{x}} LCB\big(\boldsymbol{x}|\ \mathcal{GP}_{\boldsymbol{S}_n}\big)$ ▷ Get next simulation (or control) parameter vector
  $\boldsymbol{\mu}_1 \leftarrow \boldsymbol{x}_n;\ \sigma_1 \leftarrow 1.0;\ \boldsymbol{C}_1 \leftarrow \boldsymbol{I}$ (identity); K=10 ▷ Initialize local population distribution
  for $i = 1..local_steps$ **do**
    $\boldsymbol{x}_k \sim \mathcal{N}\big(\boldsymbol{\mu}_i, \sigma_i^2\boldsymbol{C}_i\big), k = 1..K$ ▷ Sample K population candidates
    $l_k \leftarrow Sim(\boldsymbol{x}_k), k = 1..K$ ▷ Compute losses l_k with $\boldsymbol{x}_k$ as sim/control parameters
    $K_{best} \leftarrow sort(l_k)$ ▷ Get candidates with the lowest loss
    $\boldsymbol{s}_1 = \frac{1}{|K_{best}|}\sum_{k\in K_{best}}\boldsymbol{x}_k$ ▷ Compute descent start point (CMA-ES population mean)
    for $j = 1..J$ **do**
      $l_k, \nabla_{Sim}|_{\boldsymbol{s}_j} \leftarrow Sim(\boldsymbol{s}_j)$ ▷ Compute sim. loss l_j and gradients $\nabla_{Sim}|_{\boldsymbol{s}_j}$
      $\boldsymbol{s}_j \leftarrow \boldsymbol{s}_{j-1} - \alpha\nabla_{Sim}|_{\boldsymbol{s}_j}$ ▷ Take a gradient step
      Break if l_j stagnates for more than 3 steps
    $\boldsymbol{\mu}_{i+1} \leftarrow \boldsymbol{s}_J;\ \sigma_{i+1}, \boldsymbol{C}_{i+1} \leftarrow$ Eq.14-17 from [29] ▷ Update local population distribution
    $\boldsymbol{S}_{n+1} = \boldsymbol{S}_n \cup \{(\boldsymbol{s}_j, l_j)\}_{j=1}^{J} \cup \{(\boldsymbol{x}_k, l_k)\}_{k=1}^{K}$ ▷ Update data for GP posterior

avoid wasting computation when stuck on a plateau, our method monitors the quality of the gradient-based evolution of the mean and terminates unpromising descents early. BO-Leap operates on three levels: global, semi-local and local (gradient descent), which allows it to tackle loss landscapes that are challenging due to various aspects: local optima, non-smooth losses, and noisy gradients. In the next section, we show that BO-Leap has strong empirical performance in contact-rich scenarios with highly deformable objects, plastic materials and liquid.

4 A Suite of Differentiable Simulation Scenarios

Framework	Simulation Type	Supported Models	Environments
Nimble [14]	Mesh-based	Rigid	Cartpole, 3-Link Cartpole*, Pinball*
Warp [27]	Multiple	Rigid, deformable, fluid	Bounce*
DiffTaichi [19]	Multiple	Rigid, deformable, fluid	Fluid*, Swing*, Flip*
PlasticineLab [2]	Particle-based	Rigid, deformable	Assembly, Pinch, RollingPin, Rope, Table, Torus, TripleMove, Writer

Table 1: Summary of differentiable simulation frameworks and environments we use in this work. We used several existing environments and created new ones (marked with asterisks *). See supplement for more details.

In this work, we implement scenarios using several differentiable simulation frameworks. Table 1 summarizes their properties. Our main goal is to study scenarios with deformables, because prior works already explored a number of tasks limited to rigid-body motion (see previous section). That said, very few works considered contact-rich tasks, and one prior work highlighted the potential fundamental limitations of gradients for rigid contacts [6]. Hence, there is a need to further study the fundamentals of how loss landscapes and gradient fields are affected by rigid contacts. For that, we created several scenarios using Nimble [14] and Warp [27] frameworks (described below).

3-Link Cartpole: an extension of the classic Cartpole to get more challenging dynamics. Here, a 3-link pole needs to reach the blue target with its tip. Cart velocity and joint torques are optimized for each of the 100 steps of the episode, yielding a 400-dimensional optimization problem.

Pinball and Bounce: a ball launches and bounces off colliders as in a pinball game. We optimize orientations of n colliders to route the ball from the top to the blue target at the bottom. *Pinball* helps analyze effects of increasing the number of contacts, implemented in Nimble [14]. We also created a simplified *Bounce* scenario to study effects of a single collision in Warp [27], which produced well-behaved gradients in prior work [23].

To go beyond rigid objects, mesh-based simulations can model highly deformable objects, such as cloth. Particle-based simulators can model interactions with granular matter and liquids, plastic deformation with objects permanently elongating, buckling, bending, or twisting under stress. We created several mesh-based and particle-based environments that involve deformables using DiffTaichi [19] and compared the quality of the loss landscapes and gradient fields they yield. We also included environments from the PlasticineLab [2] that focus on plastic deformation.

Fluid: a particle-based fluid simulation that also involves two rigid objects (a spoon and a sugar cube) interacting in fluid. The objective is to scoop the sugar cube out of the liquid. We optimize forward-back and up-down velocity of the spoon, and let it change fives times per episode, yielding a 10-dimensional optimization problem.

Assembly, Pinch, RollingPin, Rope, Table, Torus, TripleMove, Writer: scenarios based on PlasticineLab [2] with particle-based simulations of plastic deformation. We optimize 3D velocities of anchors or pins, allowing them to change five times per episode. This yields a 90D problem for *TripleMove*, 30D for *Assembly*, 15D for the rest.

Swing: a basic cloth swinging scenario to study dynamic tasks with deformables. We give options to optimize the speed of the anchor that swings the cloth, the cloth width & length, and stiffness of cloth that is partitioned into $n \times m$ patches. Varying n, m lets us experiment with effects of increasing dimensionality of the optimization problem.

Flip: a scenario with highly dynamic motion and rigid-deformable object collisions. The goal is to flip a pancake by moving the pan. We optimize pan motion: n waypoints for left-right, up-down position and pan tilt. The pancake is modeled using a mass-spring model with small stiffness to avoid large forces from dynamic movement.

We built this suite of environments to be representative of a wide range of possible robot manipulation scenarios, regardless of whether differentiable simulators produce sensible gradients in them. We found that some of these environments yield high-quality gradients, while others do not. In Section 5.1, we show that *Cartpole*, *Fluid*, and the eight PlasticineLab environments produce well-behaved gradients and that our method can outperform competing baselines in these scenarios. Then, in Section 5.3, we show that differentiable simulators can produce incorrect gradients in highly dynamic and contact-rich environments like *Pinball*, *Swing*, and *Flip*, which makes gradient-based methods (including our method) less effective in these environments.

5 Experiments and Analysis of Optimization Landscapes

In this section, we present visual analysis of optimization landscapes and gradients, as well as comparison experiments, in various environments. We use Rand (random search) and CMA-ES as our gradient-free baselines; we use RandDescents, an algorithm that runs multiple gradient descents with randomly sampled initial parameter value, and BO as our gradient-based baselines.

5.1 Simulation Experiments and Analysis

We first analyze using gradients with rigid objects. The left side of Figure 2 shows experiments on the *1-Link Cartpole*, where gradient-free CMA-ES performs similar to gradient-based algorithms (RandDescents, BO), while BO-Leap obtains a significantly lower loss. The right side shows results for *3-Link Cartpole* which has much more complex dynamics. Here, all gradient-based algorithms

show a large improvement over gradient-free ones. CMA-ES does better than a completely random search (Rand), but fails to lift the pole tip to the target.

On the top-right corner of Figure 2, we show qualitative results from a CMA-ES run (top) and from a BO-Leap run (bottom), where BO-Leap brings the tip close to the target.

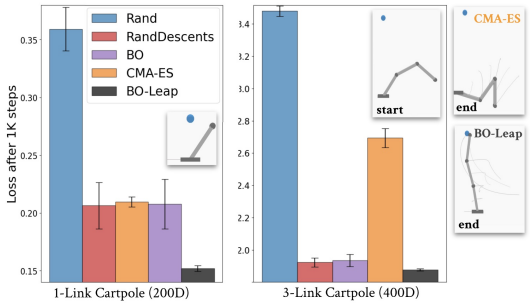

Figure 2: Left: results for *1-Link Cartpole*. Right: results for *3-Link Cartpole*. For all bar charts in this paper, the vertical axis shows the mean and 95% confidence interval of best loss value after 1,000 optimization steps. Each optimization step runs one simulation episode, computes the loss, and propagates the gradient wrt. optimization parameters if the algorithm needs it.

Next, we study a variety of scenarios that use particle-based simulation. Particle simulators have much larger computational and memory requirements than mesh-based options. Gradient computations further increase resource requirements. For the differentiability to be warranted, benefits of gradients have to be significant. We show that gradient-based methods can indeed have large benefits. Figure 3 visualizes the *Fluid* scenario: the left side shows a 2D slice of the loss landscape. It has many valleys with shallow local optima and appears more difficult than many test landscapes designed to challenge global optimization methods. The middle of Figure 3 shows directions of gradients produced by the differentiable simulator. The right side shows results for gradient-based and gradient free methods. CMA-ES gets stuck in a local optimum: the spoon fails to lift the sugar cube in most runs as shown in the next-to-last column. In contrast, BO-Leap successfully lifts the sugar cube in most runs, outperforming all baselines. BO is designed to balance allocating trials to global exploration while still reserving enough trials to return to the well-explored regions that look promising. In this scenario such global optimization strategy proves to be beneficial. Furthermore, BO-Leap also handles the rough parts of the landscape more effectively than the baseline BO.

In the next set of experiments, we analyze eight environments from PlasticineLab [2]. Figure 4 shows a *Rope* scenario, where the objective is to wrap it around a rigid cylindrical pole. The loss landscape is smooth in most dimensions (the left plot shows an example 2D slice). However, higher dimensionality makes the overall problem challenging. The middle plot confirms that gradients are correct in most parts, but also shows a large plateau where even the gradient-based approaches are likely to get stuck. The right side shows evaluation results: CMA-ES and BO fail to wrap the rope fully around the pole, while BO-Leap succeeds in pulling the ends on the back side of the pole correctly. This environment also shows that simply using gradients with random restarts is not sufficient even on smooth landscapes: RandDescents has poor performance that does not improve significantly over gradient-free random search.

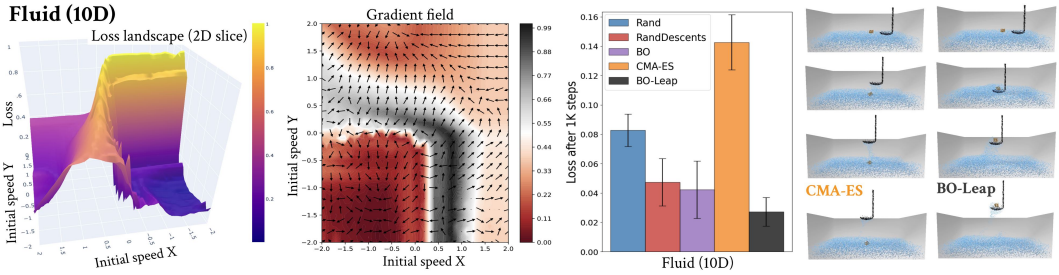

Figure 3: A scenario with scooping up a sugar cube from fluid. The left side shows a 2D slice of the 10D optimization landscape and the corresponding gradients. To make gradient directions visible, we normalize the magnitude of gradients in all gradient plots in this paper; the arrows point towards the direction of negated gradients (i.e. the direction gradient descent updates take). The right plot shows quantitative evaluation of optimization methods. We visualize qualitative results for CMA-ES and BO-Leap on the right side.

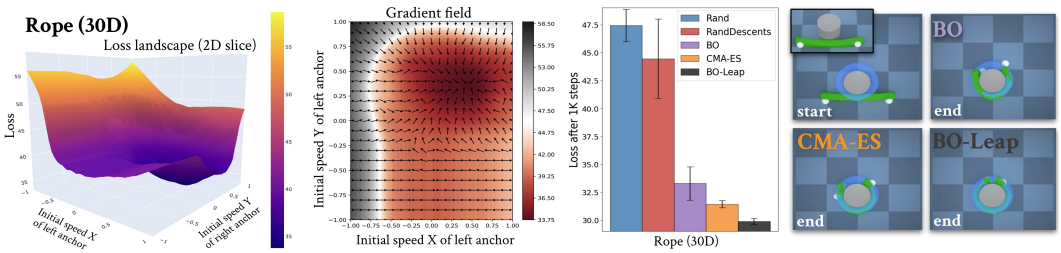

Figure 4: Results for the *Rope* environment. BO-Leap outperforms all other methods and successfully completes the task of wrapping the rope fully around the pole (light blue region shows the target shape). Gradient-free CMA-ES cannot reliably get the optimal behavior. While BO-Leap uses gradients effectively, this scenario shows that simply using gradients with random restarts is not sufficient: RandDescents has poor performance.

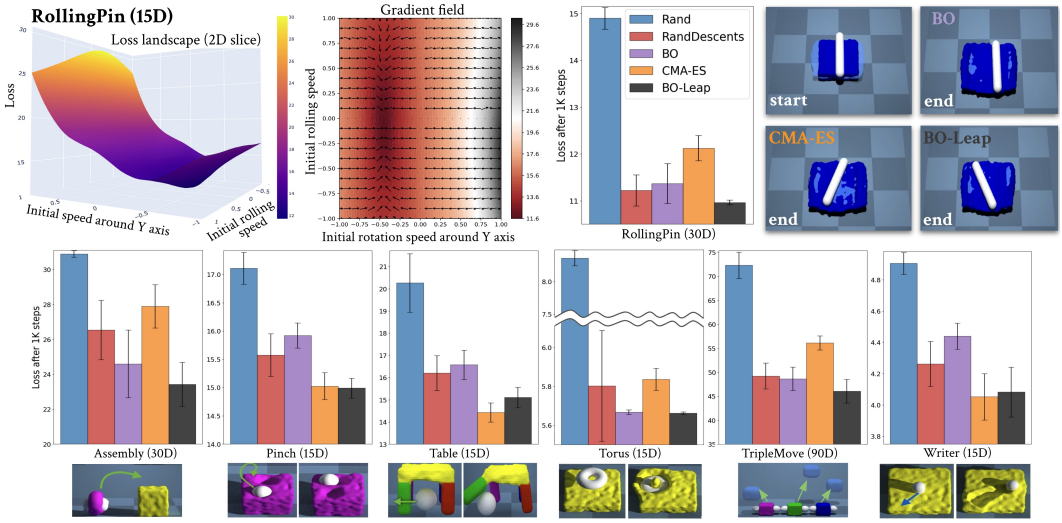

Figure 5: Top row: analysis and results for *RollingPin* task. Bottom rows: results for the other six PlasticineLab tasks. Plots show mean performance over 10 runs of each method per task (see supplemental for further details). Environment illustrations below the six bar charts at the bottom of this figure are borrowed from [2].

Figure 5 shows experiments with the other seven PlasticineLab tasks. The top shows the *RollingPin* task, where the objective is to spread the dark-blue dough using a thin rigid cylindrical white pin (the light-blue region shows the target shape). The loss landscape is smoother than that of the *Rope* task, but has even larger plateaus and flat gradients in most regions, making the problem challenging despite a smooth loss. BO-Leap outperforms CMA-ES and gradient-based methods on this task as well. Example frames on the right show that CMA-ES thins out the dough too much (large gaps appear in the middle, revealing the light-blue target shape underneath). BO and BO-Leap keep the central portion of the dough more uniformly spread. The bottom plots show results for the additional six PlasticineLab tasks. BO-Leap outperforms gradient-based methods in all tasks, and achieves significantly lower loss than CMA-ES in *Assembly*, *Torus*, and *TripleMove* tasks (see supplementary materials for further details).

5.2 Validation in Real Robot Setup

To validate the proposed method using real data, we consider the task of identifying the properties of a simulated deformable object to make its motion match the real motion. In this scenario, a Gen3 (7DoF) Kinova robot manipulates a small deformable object by lifting it up from the table surface. The object is tracked by two RealSense D435 depth cameras, one placed overhead, the other on the side. The objective is to optimize the size (width & length), mass, and friction of the deformable object, as well as stiffness of each of the $8 \times 8 = 64$ patches of the object, which yields a 68D optimization problem. The loss penalizes the distance of the simulated corner vertices to the position of the corners marked on the real object. Figure 6 shows results of experiments with two

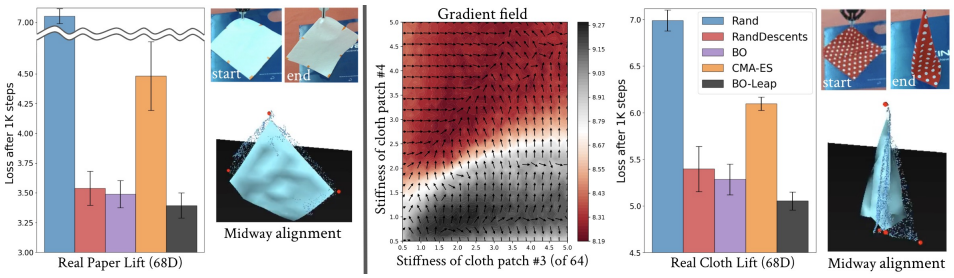

Figure 6: Results for optimizing a simulated object to match the motion of a real paper (left) and cloth (right), which are lifted by a real robot. We visualize the midway alignment showing that our method finds the size, mass, friction and stiffness (for each of the 64 patches) to bring simulated object behavior close to the real one.

objects: a stiff but flexible paper (left) and a highly flexible cloth (right). In both cases, BO-Leap is able to find physical simulation parameters that produce the best alignment of the simulated and real object. This offers a validation of the method on real data, and shows that it can tackle 'real2sim' problems to automatically bring the behavior of simulated objects closer to reality. This is valuable for highly deformable objects, since manual tuning is intractable for high dimensions.

5.3 Limitations

Modeling global loss posterior with BO can be computationally expensive. We use BoTorch [35] for BO on GPU, which can scale to high dimensions. We focused on results within a budget of 1K optimization steps. If a much larger budget is allowed, then more tests would be needed to validate that BoTorch (or other frameworks) can scale well in terms of compute and memory resources.

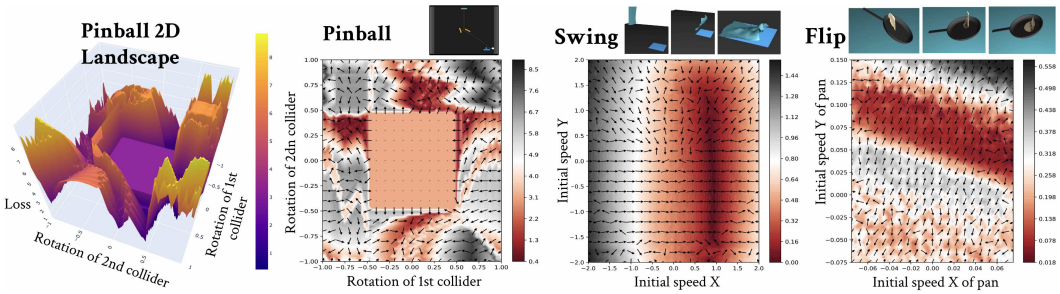

Figure 7: Visual insights into the challenges of obtaining well-behaved gradients for cases with rigid contacts in *Pinball*, and for deformables in the presence of contacts and highly dynamic tasks, such as *Swing* & *Flip*.

The biggest challenges of optimization with differentiable simulators arise due to quality of the gradients, which can be insufficient to be beneficial for gradient-based algorithms, including our method. In Figure 7, we show three environments where gradients produced by differentiable simulators are of poor quality. In the *Pinball* environment, gradients with respect to collider orientations are computable only if a collision (with the pinball) has occurred to begin with. In addition to collision-induced discontinuities, the absence of gradients results in plateaus, affecting gradient-based optimizers. Even in a state-of-the-art simulator Warp [27] with relaxed contact models, a simple *Bounce* task induces gradient discontinuities (see supplement for details). In *Swing* and *Flip* tasks, while the dynamics appear realistic, differentiable simulators yield gradients with incorrect (often opposite) directions. This is a common pitfall for practitioners who use differentiable simulators without assessing loss landscapes first.

Conclusion Our analysis shows that differentiable simulation of contact-rich manipulation scenarios results in loss landscapes that are difficult for simple gradient-based optimizers. To overcome this, we proposed a hybrid approach that combines local (gradient-based) optimization with global search, and demonstrated success on rugged loss landscapes, focusing on cases with deformables. We believe our analyses and tools provide critical feedback to differentiable simulator designers and users alike, to take differentiable simulators a step closer to real-world robot learning applications.

References

[1] K. M. Jatavallabhula, M. Macklin, F. Golemo, V. Voleti, L. Petrini, M. Weiss, B. Considine, J. Parent-Levesque, K. Xie, K. Erleben, L. Paull, F. Shkurti, D. Nowrouzezahrail, and S. Fidler. gradsim: Differentiable simulation for system identification and visuomotor control. 2021.

[2] Z. Huang, Y. Hu, T. Du, S. Zhou, H. Su, J. B. Tenenbaum, and C. Gan. PlasticineLab: A soft-body manipulation benchmark with differentiable physics. 2021.

[3] X. Lin, Z. Huang, Y. Li, J. B. Tenenbaum, D. Held, and C. Gan. Diffskill: Skill abstraction from differentiable physics for deformable object manipulations with tools. *arXiv preprint arXiv:2203.17275*, 2022.

[4] P. Sundaresan, R. Antonova, and J. Bohg. DiffCloud: Real-to-Sim from Point Clouds with Differentiable Simulation and Rendering of Deformable Objects. *arXiv preprint arXiv:2204.03139*, 2022.

[5] P. Ma, T. Du, J. B. Tenenbaum, W. Matusik, and C. Gan. Risp: Rendering-invariant state predictor with differentiable simulation and rendering for cross-domain parameter estimation. *arXiv preprint arXiv:2205.05678*, 2022.

[6] H. J. T. Suh, M. Simchowitz, K. Zhang, and R. Tedrake. Do differentiable simulators give better policy gradients? *arXiv preprint arXiv:2202.00817*, 2022.

[7] J. Degrave, M. Hermans, J. Dambre, et al. A differentiable physics engine for deep learning in robotics. *Frontiers in neurorobotics*, page 6, 2019.

[8] F. de Avila Belbute-Peres, K. Smith, K. Allen, J. Tenenbaum, and J. Z. Kolter. End-to-end differentiable physics for learning and control. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems (NeurIPS)*. Curran Associates, Inc., 2018.

[9] C. Song and A. 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, 10–11 Jun 2020.

[10] C. Song and A. Boularias. Learning to slide unknown objects with differentiable physics simulations. *arXiv preprint arXiv:2005.05456*, 2020.

[11] Y.-L. Qiao, J. Liang, V. Koltun, and M. C. Lin. Efficient differentiable simulation of articulated bodies. In *International Conference on Machine Learning*, pages 8661–8671. PMLR, 2021.

[12] M. Lutter, J. Silberbauer, J. Watson, and J. Peters. Differentiable physics models for real-world offline model-based reinforcement learning. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4163–4170. IEEE, 2021.

[13] M. A. Z. Mora, M. P. Peychev, S. Ha, M. Vechev, and S. Coros. Pods: Policy optimization via differentiable simulation. In *International Conference on Machine Learning*, pages 7805–7817. PMLR, 2021.

[14] K. Werling, D. Omens, J. Lee, I. Exarchos, and C. K. Liu. Fast and feature-complete differentiable physics engine for articulated rigid bodies with contact constraints. In *Robotics: Science and Systems*, 2021.

[15] M. Geilinger, D. Hahn, J. Zehnder, M. Bächer, B. Thomaszewski, and S. Coros. Add: Analytically differentiable dynamics for multi-body systems with frictional contact. *ACM Transactions on Graphics (TOG)*, 39(6):1–15, 2020.

[16] A. H. Taylor, S. Le Cleac'h, Z. Kolter, M. Schwager, and Z. Manchester. Dojo: A differentiable simulator for robotics. *arXiv preprint arXiv:2203.00806*, 2022.

[17] Y.-L. Qiao, J. Liang, V. Koltun, and M. Lin. Scalable differentiable physics for learning and control. In *International Conference on Machine Learning*, pages 7847–7856. PMLR, 2020.

[18] C. D. Freeman, E. Frey, A. Raichuk, S. Girgin, I. Mordatch, and O. Bachem. Brax - a differentiable physics engine for large scale rigid body simulation. In *Conference on Neural Information Processing Systems (NeurIPS) Datasets and Benchmarks Track*, 2021. URL https://openreview.net/forum?id=VdvDlnnjzIN.

[19] Y. Hu, L. Anderson, T.-M. Li, Q. Sun, N. Carr, J. Ragan-Kelley, and F. Durand. Difftaichi: Differentiable programming for physical simulation. 2020.

[20] Y. Hu, J. Liu, A. Spielberg, J. B. Tenenbaum, W. T. Freeman, J. Wu, D. Rus, and W. Matusik. Chainqueen: A real-time differentiable physical simulator for soft robotics. In *2019 International conference on robotics and automation (ICRA)*, pages 6265–6271. IEEE, 2019.

[21] J. Liang, M. Lin, and V. Koltun. Differentiable cloth simulation for inverse problems. *Advances in Neural Information Processing Systems*, 32, 2019.

[22] N. Thuerey. Differentiable fluid simulations for deep learning. In *APS Division of Fluid Dynamics Meeting Abstracts*, pages H17–006, 2019.

[23] E. Heiden, M. Macklin, Y. S. Narang, D. Fox, A. Garg, and F. Ramos. DiSECt: A Differentiable Simulation Engine for Autonomous Robotic Cutting. In *Proceedings of Robotics: Science and Systems*, Virtual, July 2021. doi:10.15607/RSS.2021.XVII.067.

[24] W. Wang, S. Axelrod, and R. Gómez-Bombarelli. Differentiable molecular simulations for control and learning. 2020.

[25] P. Ma, T. Du, J. Z. Zhang, K. Wu, A. Spielberg, R. K. Katzschmann, and W. Matusik. Diffaqua: A differentiable computational design pipeline for soft underwater swimmers with shape interpolation. *ACM Transactions on Graphics (TOG)*, 40(4):132, 2021.

[26] S. S. Schoenholz and E. D. Cubuk. Jax m.d. a framework for differentiable physics. In *Advances in Neural Information Processing Systems*, volume 33. Curran Associates, Inc., 2020. URL https://papers.nips.cc/paper/2020/file/83d3d4b6c9579515e1679aca8cbc8033-Paper.pdf.

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

[28] M. Locatelli and F. Schoen. (global) optimization: Historical notes and recent developments. *EURO Journal on Computational Optimization*, 9:100012, 2021.

[29] N. Hansen and A. Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evolutionary computation*, 9(2):159–195, 2001.

[30] N. Hansen. The cma evolution strategy: a comparing review. *Towards a new evolutionary computation*, pages 75–102, 2006.

[31] N. Hansen, A. Auger, R. Ros, S. Finck, and P. Pošík. Comparing results of 31 algorithms from the black-box optimization benchmarking bbob-2009. In *Proceedings of the 12th annual conference companion on Genetic and evolutionary computation*, pages 1689–1696, 2010.

[32] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. de Freitas. Taking the Human Out of the Loop: A Review of Bayesian Optimization. *Proceedings of the IEEE*, 104(1):148–175, 2016.

[33] X. Chen, X. Liu, and Y. Jia. Combining evolution strategy and gradient descent method for discriminative learning of bayesian classifiers. In *Proceedings of the 11th Annual conference on Genetic and evolutionary computation*, pages 507–514, 2009.

[34] C. K. Williams and C. E. Rasmussen. *Gaussian processes for machine learning*, volume 2. MIT press Cambridge, MA, 2006.

[35] M. Balandat, B. Karrer, D. R. Jiang, S. Daulton, B. Letham, A. G. Wilson, and E. Bakshy. BoTorch: A Framework for Efficient Monte-Carlo Bayesian Optimization. In *Advances in Neural Information Processing Systems 33*, 2020. URL http://arxiv.org/abs/1910.06403.