A Tale of Two Gradients: Exact and Stochastic Approaches to Optimization through Contact

Charles Dawson
Dept. of Aeronautics and Astronautics
Massachusetts Institute of Technology
Cambridge, USA
cbd@mit.edu

***Abstract*—The optimization problems that arise when planning for manipulation tasks can exhibit sharp peaks and local minima — features that are traditionally very difficult for exact gradient-based methods. As a result, recent years have seen many works successfully applying stochastic gradient-free optimization methods to manipulation problems. Moreover, the difficulty of obtaining analytical gradients for contact dynamics (and the potentially discontinuous nature of those gradients) provide an additional advantage for gradient-free methods like reinforcement learning. Despite this intuitive advantage for gradient-free methods, we design a controlled experimental comparison to show that exact gradient-based methods in fact outperform stochastic methods on simplified manipulation tasks — successfully avoiding local minima to discover feasible manipulation strategies. These results suggest that although optimization problems in manipulation can be "nasty" in the sense of a poorly behaved cost landscape, this nastiness does not practically affect the performance of exact gradient-based methods.**

***Index Terms*—optimization, manipulation, automatic differentiation**

I. Introduction

To `grad` or not to `grad`, that is the question. At risk of mixing literary metaphors, this project explores a tale of two gradients: when it comes to solving manipulation problems using optimization, is it better to use exact gradients or rely on stochastic gradient-free methods? Conventional wisdom in optimization would suggest that manipulation problems, with numerically stiff dynamics and cost functions and many local minima, are best solved by stochastic methods [1]. Moreover, the difficulty of deriving analytical gradients for contact dynamics has ceded ground to gradient-free methods like reinforcement learning, which use randomness to explore the search space and find optimal solutions. However, the emergence of easy-to-use automatic differentiation software means that it is now easy to obtain exact gradients, even for complex dynamics such as contact and frictional forces. This begs the question: can exact gradients compete with stochastic gradient-free methods for solving optimization problems arising from manipulation?

A. Related Work

1) Gradient-based Optimization for Contact: A number of previous works have applied gradient-based optimization to solve planning problems with contact [2]–[4]. These works generally take one of two approaches. [3], [4] use analytical gradients to linearize the contact dynamics and solve a sequence of convex optimization problems (iterative LQR, or iLQR, in [4] and sequential quadratic programming, or SQP, in [3]). These works consider the dynamics separately from the cost function. Another line of work [2] uses the end-to-end gradient of the cost function with respect to the decision variables, but these works typically relax the contact dynamics and use finite differences to approximate this end-to-end gradient. In this work, we consider the end-to-end gradient, like [2], and compare two methods of computing this gradient (an exact method using automatic differentiation and an approximate stochastic method).

2) Gradient-free Optimization for Contact: In parallel to these gradient-based planning methods, a large body of work has demonstrated the success of gradient-free methods for planning through contact, most notably reinforcement learning (RL). Some methods in this area learn deep models of contact dynamics and then optimize trajectories using this model in a gradient-free manner [5]. Other methods take a model-free approach and use gradient-free optimization methods directly on the task reward [6]. Research in this direction has generated whole libraries of gradient-free optimization strategies, from specific model-free RL algorithms [7] to more general purpose optimizers [8].

Generally speaking, gradient-free methods operate stochastically, adding random exploration to the optimization process. The intuition is that this stochasticity helps explore the search space and escape local minima [1]. Viewed through this lens, it is not surprising that these methods have worked well for manipulation-based optimization problems, which we expect to be stiff (in the sense of having regions of relatively flat cost near regions of sharply varying cost) and have multiple local minima [4].

3) Automatic Differentiation: Some of the success of gradient-free methods like RL can be attributed to their substantial ease of use. Historically, it has been much easier to write a black-box simulator for some robotics problem than to write a simulator that provides the gradients (as demonstrated informally by the proliferation of OpenAI gym environments in RL research). Given sufficient manual effort, it is of course possible to analytically derive the gradients of contact events [9], [10], but this requires additional engineering effort.

Recently, automatic differentiation has received some attention as an alternative to manually specifying analytical gradients, with tools such as Taichi [11] and JAX [12] enabling automatically differentiable simulators to be written with relatively little engineering overhead. These methods impose additional computational overhead, but this can be partially mitigated by combining automatic differentiation with just-in-time compilation for fast execution.

B. Contributions

With the advent of new automatic differentiation tools, the ease-of-use of gradient-free methods becomes a less compelling argument for their advantage over gradient-based methods: an automatically differentiable OpenAI gym environment can now be written in the same amount of time as a black-box environment. Once this factor is removed, we are left with the question: if exact gradients are available for manipulation tasks, when would we choose to use these exact gradients instead of (typically stochastic) gradient-free methods? Generally accepted wisdom [1] would hold that stochastic algorithms will perform better on the stiff, multi-modal optimization problems typical of manipulation than gradient-based optimization, but little work has been done in conducting controlled experiments to compare the performance of these two types of optimization scheme on manipulation problems.

To fill this gap, this paper provides an empirical comparison between gradient-based and gradient-free optimization methods for solving manipulation problems. This paper uses deliberately simplified optimization algorithms: gradient descent using exact gradients and gradient descent using gradients approximated with stochastic perturbations. Our goal here is not to advance the state of the art in terms of what can be done with optimization for manipulation, but to advance the state of our understanding about when exact gradients can and cannot be used for these optimization problems.

By conducting a controlled comparison of these two optimization methods, we show that gradient descent with exact gradients performs *better* than gradient-free stochastic descent on the manipulation problems we consider. Despite "nasty" cost landscapes (containing multiple local minima and flat regions bordering sharply-varying regions), the exact gradients are able to not only converge faster than stochastic methods, but in our more complicated example succeed where the stochastic method fails (even when run for 10 times longer than the exact method). Combined with recent advances in automatic differentiation [11], [12], our results provide new hope that gradient-based optimization still has much to offer the field of manipulation, and that gradient-free optimization may not always be the best choice for this class of problem.

II. Methodology

This work seeks to answer the question, "do exact gradients provide any benefits over stochastic gradient approximations for optimization control policies for manipulation tasks?" To answer this question, we must isolate (as much as possible) effects from the gradient from other effects, such as choice of optimization algorithm or cost function. This section describes our approach to setting up the experimental comparison used in this paper. All code required to reproduce our experiments can be found at github.com/dawsonc/a_tale_of_two_gradients.

A. Simulator

We implemented two simple manipulation tasks: a box flipping task and a box grasping task, shown in Fig. 1. The robot is represented as a point subject to gravity and controlled via stiffness control relative to a desired location (x_{des}, z_{des}) (in the grasping task, the robot is represented as two separately controlled points). The robot interacts via contact with an unactuated box. The robot dynamics are:

$$m_r \ddot{x}_r = -k_p(x_r - x_{des}) - k_p \dot{x}_r + f_n \hat{n} \cdot \hat{x} + f_t \hat{t} \cdot \hat{x} \tag{1}$$

$$m_r \ddot{z}_r = -m_r g - k_p(z_r - z_{des}) - k_p \dot{z}_r + f_n \hat{n} \cdot \hat{z} + f_t \hat{t} \cdot \hat{z} \tag{2}$$

where $m_r = 0.1\,\mathrm{kg}$ is the mass of the robot finger, g is the gravitational constant, k_p and $k_d = 2\sqrt{m_r k_p}$ are stiffness control constants ($k_p = 10\,\mathrm{N/m}$ for the flipping task and $40\,\mathrm{N/m}$ for the grasping task), and f_n and f_t are the contact forces between the robot and the box in the normal $\hat{n}$ and tangent $\hat{t}$ directions. The box has similar dynamics, but without the control terms and with additional contact forces between each corner and the ground (the box has mass $1\,\mathrm{kg}$ and side length $0.5\,\mathrm{m}$). Contact is modelled using the penalty method as described in [4], where the normal and tangent forces are given by

$$f_n = k_c \min(\phi, 0) \tag{3}$$

$$f_t = \mu f_n \tag{4}$$

$$\mu = \begin{cases} c\psi & \psi \leq \psi_s \\ \mu_d & \psi > \psi_s \end{cases} \tag{5}$$

where ϕ is the signed distance between the two colliding bodies, k_c is a stiffness constant ($1000\,\mathrm{N/m}$), ψ is the relative tangential velocity between the two bodies, $\mu_d = 0.7$ is the coefficient of dynamic friction, $c = 2.0\,\mathrm{s/m}$ is a scaling parameter governing static friction, and $\psi_s = \mu_d / c$ is set to make the Coulomb friction model continuous. These rigid body and contact dynamics were implemented in Python using the JAX framework [12], which allows for easy automatic differentiation. When exact gradients are used, they are obtained using the JAX `value_and_grad` function, which automatically differentiates the cost function and the underlying dynamics simulator.

B. Problem specification

The objectives in the flipping and grasping tasks were specified using a cost function, which was the same for both optimization methods. These cost functions are purposefully sparse to minimize the possibility for reward engineering. In both cases, the cost is a function of the final state of the box $[x, z, \theta, \dot{x}, \dot{z}, \dot{\theta}]_T$ at the end of simulating for a fixed number of timesteps (600 steps of $1\,\mathrm{ms}$ each for the flipping task, 500

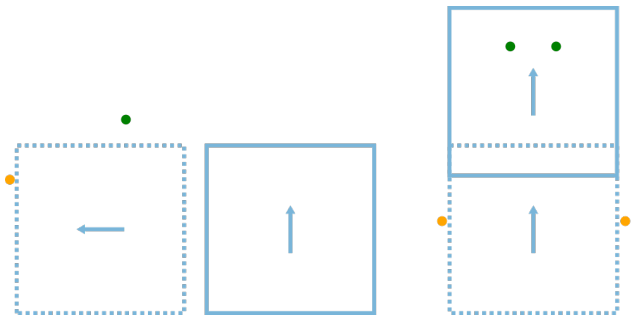

Fig. 1. The box flipping and box grasping tasks (left and right, respectively), showing the starting configuration (dashed blue) and goal configuration (solid blue). The decision variables are the (x, z) locations of the setpoint (green) for each finger (orange).

steps for the grasping task). The cost for each task is given below.

$$J_{flip} = x_T^2 + \theta_T^2 \tag{6}$$

$$J_{grasp} = (y_T - 1)^2 + 0.1(x_T^2 + \theta_T^2 + \dot{x}_T^2 + \dot{y}_T^2 + \dot{\theta}_T^2) \tag{7}$$

The initial conditions of the box is $[-1, 0.25, \pi/2, 0, 0, 0]$ for the flipping task and $[0, 0.25, 0, 0, 0, 0]$. The goal in both tasks is to find setpoints x_{des} and z_{des} for the stiffness controller of each robot finger to minimize the cost (evaluated on the output of the simulator). For simplicity, the setpoints were treated as constant; this was sufficient to successfully accomplish both tasks. In the flipping task, all optimizations were initialized with a setpoint that caused the robot to make contact and push the box without flipping it. In the grasping task, all optimizations were initialized with setpoints that made contact with the box but did not move it. These initial guesses were not tuned by hand to achieve good performance; they were simply chosen as sensible defaults.

Note that this perspective on gradient-based optimization for manipulation differs from another common approach in the literature [3], [4]. Instead of using the gradients to linearize the dynamics and applying iterative LQR [4] or sequential quadratic programming [3], we use automatic differentiation to obtain the gradient of the cost with respect to the decision variables directly, optimizing through the contact simulator. This makes our approach similar in spirit to [2], but we do not relax contact dynamics beyond using the penalty approximation, and we compute exact gradients using automatic differentiation rather than finite differences.

C. Optimization methods

To minimize the effect of the particular optimization method chosen on the results of the comparison between exact and approximate gradients, we deliberately chose a simple gradient-descent optimization method. For exact gradients, the optimization update of decision variables θ is given by

$$\theta_{i+1} = \theta_i - \alpha_1 \nabla J(\theta_i) \tag{8}$$

Task	Parameter	Value
Flip	α_1	0.1
	α_2	0.1
	σ	0.5
Grasp	α_1	0.5
	α_2	0.5
	σ	0.1

TABLE I
HYPERPARAMETERS FOR EACH TASK FOUND BY VIA MANUAL TUNING.

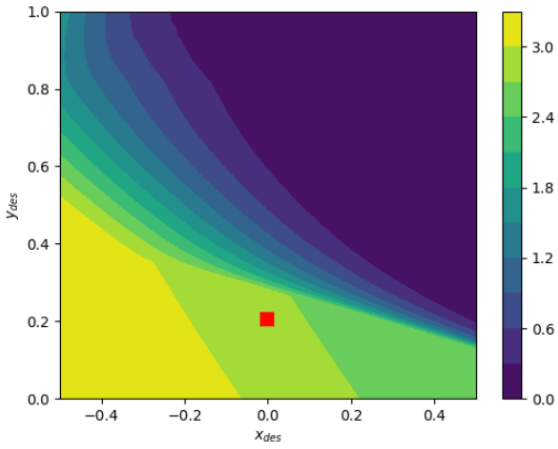

Fig. 2. The cost landscape of the box flipping task, with the initial guess marked in red.

For approximated gradients, the update rule is given by

$$\theta_{i+1} = \theta_i - \alpha_2[J(\theta_i + w) - J(\theta_i)]w \tag{9}$$

$$w \sim \mathcal{N}(0, \Sigma) \tag{10}$$

The step sizes α_1 and α_2 were tuned separately for each task, and the diagonal covariance matrix $\Sigma = \sigma I$ was also tuned for each problem. It should be clear that these two update rules are equivalent in expectation; i.e., on average, Eq. (9) steps in the same direction as Eq. (8). The only difference is in the variance of the approximate update. The hyperparameters found by manual tuning are given in Table I.

III. RESULTS

This section presents the results of applying exact gradient descent and approximate stochastic gradient descent on each manipulation task discussed in Section II; detailed discussion of these results is provided in the following section.

A. Box flipping task

For optimization problems of sufficiently low dimension, we can gain intuition about the difficulty of the problem by visualizing its cost landscape. Since the box flipping task has only two decision variables (the x and z setpoints for the robot finger), we can fully visualize the cost landscape in Fig. 2, with the initial guess marked in red.

The performance of gradient descent with both exact and approximate gradients is shown in Fig. 3. The initial behavior and optimized solutions for both methods are shown in Fig. 4.

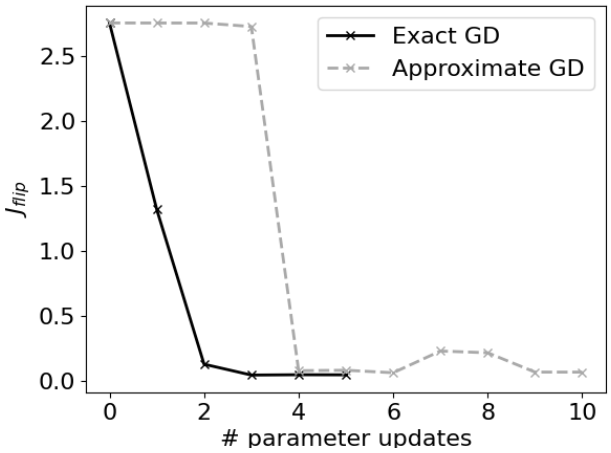

Fig. 3. The cost J_{flip} as a function of gradient descent iteration using exact and approximate gradients.

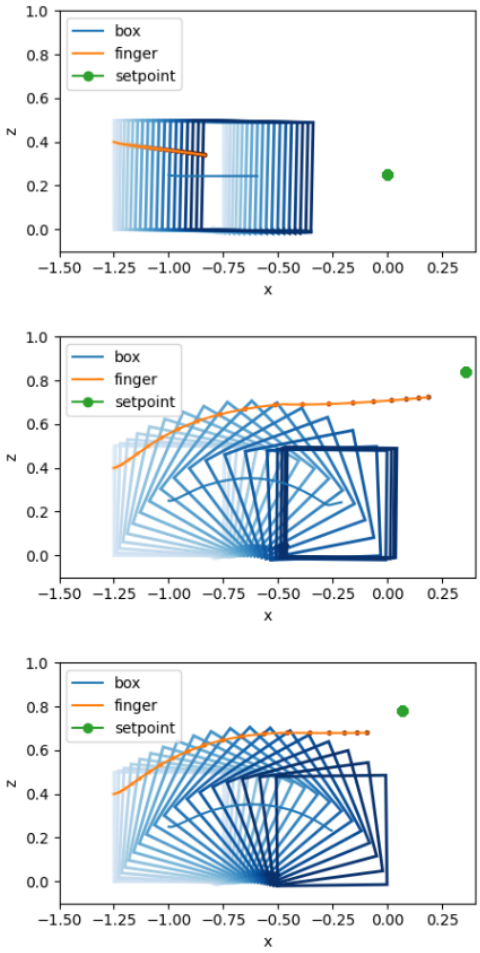

Fig. 4. (Top) The initial behavior, and the optimized behavior found using exact (middle) and approximate (bottom) gradient descent in the flipping task.

B. Box grasping task

Although the box grasping task has four decision variables, there is some symmetry to the problem that we can exploit to visualize the cost landscape. By constraining the setpoints for each finger to be symmetric about $x = 0$, we can visualize the cost landscape in Fig. 5. This constraint is not used during optimization; it is only used to enable a two-dimensional visualization. Approximate gradient descent is not guaranteed to preserve this symmetry during the solution process, but it is still useful to understand the stiffness of the optimization problem and whether it admits multiple local minima. The performance of both gradient descent variants on this task is shown in Fig. 6. The initial behavior and optimized solutions for both methods are shown in Fig. 7.

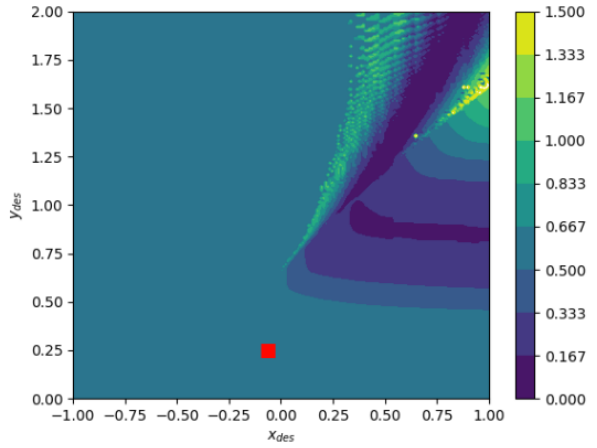

Fig. 5. The cost landscape of the box grasping task, with the initial guess shown in red.

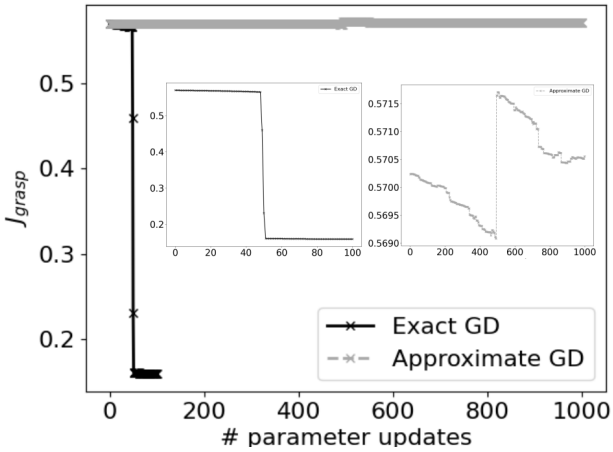

Fig. 6. The cost J_{grasp} as a function of gradient descent iteration using exact and approximate gradients. Inset figures show separate traces for exact (left inset) and approximate (right inset) gradient descent.

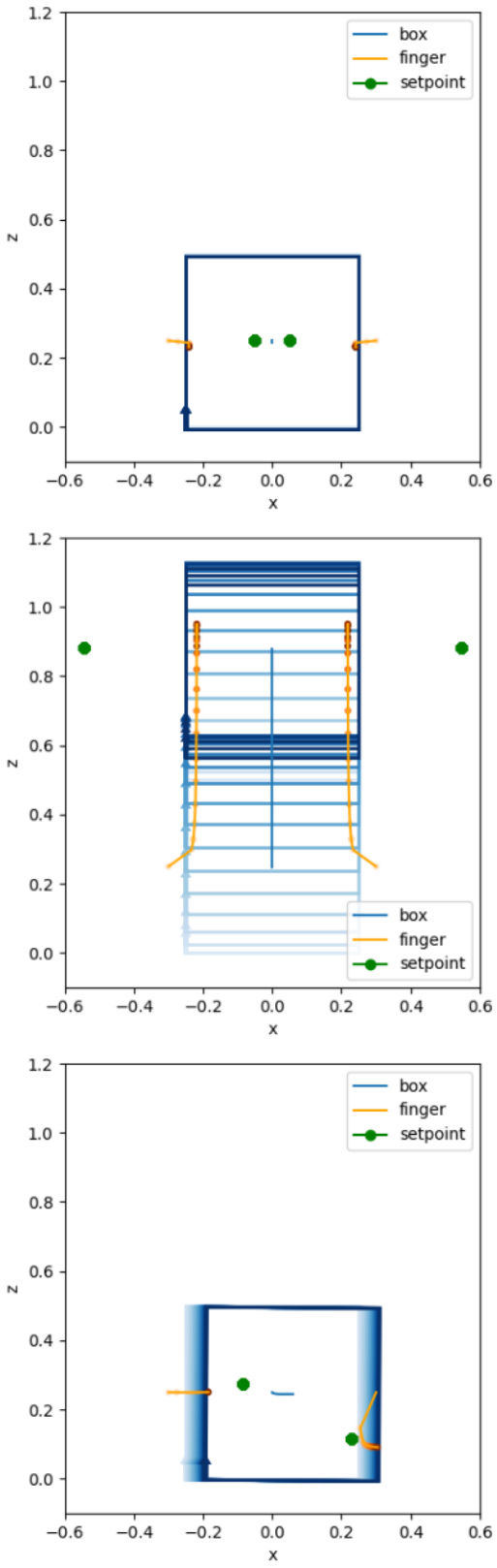

Fig. 7. (Top) The initial behavior, and the optimized behavior found using exact (middle) and approximate (bottom) gradient descent in the grasping task.

IV. Discussion

The cost landscapes of both tasks (Figs. 2 and 5) exhibit many conventionally "nasty" features for gradient descent. Both problems are stiff (in the sense of having very flat regions near regions of very sharp changes) and the grasping problem in particular has a large number of local minima. Despite these features, we find that exact gradient descent is capable of solving these problems; approximate gradient descent successfully solves only the flipping problem.

Both exact and approximate gradient descent were able to solve the flipping task, and both methods discovered similar solutions and achieved similar optimal cost. Exact gradient descent converges more quickly — requiring fewer gradient descent steps to obtain a low cost (2 rather than 4). Function evaluations with exact gradients are more expensive than those without automatic differentiation (16.76 s vs. 2.58 s for the flipping task), but the approximate method requires an additional function evaluations at each step (this extra cost could be removed by using a baseline function, but at the cost of higher variance) and tends to require more steps. Beyond this slight difference in convergence (which is hard to directly compare, since the methods' step sizes are not necessarily equivalent), both methods' performance on the box flipping task are essentially similar. Although this result may seem negative, it does provide an important, counter-intuitive insight: exact gradient descent does not appear to need random exploration to escape regions of relatively flat cost.

These methods' performance on the box grasping task provides additional insight into the relative merits of exact and approximate gradients. Exact gradient descent requires approximately 50 iterations to reach a low cost. Most of those iterations are spent learning to lift the box out of contact with the floor (where the gradients are relatively small), and once the box begins to lift convergence is relatively rapid. Surprisingly, exact gradient descent is not affected much by the regions of densely-packed spikes in the cost landscape seen in Fig. 5. In contrast, approximate gradient descent does not successfully solve this problem, even after 1000 iterations with manually tuned learning rate and perturbation strength. The most common failure mode for the approximate method was randomly exploring the region of flat cost near the initial guess (where the fingers make contact but do not lift the box) before reaching a solution where the box is moved sideways or flipped. The stochastic approximate algorithm has extreme difficulty finding the nearly symmetric lifting motion that accomplished the goal.

This failure mode points to a qualitative benefit of exact gradient descent: it can preserve symmetry in the problem. Given symmetric starting setpoints for the grasping task, the exact gradients preserve this symmetry to find a symmetric optimized solution; the stochastic approximate variant is unable to preserve this symmetry, since the random perturbations w will almost surely destroy symmetry. This effect is hard to quantify in terms of convergence rate or optimal cost, but since symmetry is present in many physical problems we might

expect this feature to bring benefits in other domains as well.

V. Conclusion and Future Work

In this work, we develop empirical insight into the relative merits of stochastic and exact gradients for optimization through contact. To do this, we attempt to solve two simplified manipulation tasks using two different variants of gradient descent: an exact version that uses automatic differentiation to compute gradients through a contact simulator, and a stochastic version that approaches exact gradient descent in expectation.

We find that although the cost landscapes of our manipulation problems are stiff and contain many sharp peaks and local minima, exact gradient descent performs better than its stochastic approximation — a surprising result, since these cost landscapes are canonically "nasty" for gradient-based optimization. We believe that the strong performance of exact gradient-based methods is due to these methods ability to respect the symmetry of physical problems, as well as the ability of the gradient to provide actionable information even in nearly flat regions of the cost landscape (in these regions, the noise from stochastic gradient-free algorithms may drown out the signal from small decreases). Although not explored in this work, gradient-based methods also enable principled treatment of constraints, which can be difficult to accommodate in a strictly gradient-free setting.

There are a few directions for future work. The focus of this work is an empirical comparison of gradient-based and gradient-free optimization methods, leaving room for a theoretical analysis to provide conditions on the cost landscape that enable strong performance from each of these methods. Such a theoretical analysis would be complementary to the experimental results presented here: we have shown *that* gradient-based methods can outpeform gradient-free methods in solving manipulation problems, it remains to show *why*.

An additional direction of future work would be to consider the role of initial conditions on the success of these optimization methods. All versions of gradient descent, but particularly those based on exact gradients, are sensitive to the initial conditions, but further work is needed to understand how that sensitivity plays out in manipulation problems. An additional line of work would be to explore a combination between exact and stochastic methods that allows for some random exploration while still preserving the speed and efficiency of exact gradients.

A. Disclosure for 6.843

As mentioned in my project proposal, this project relates not only to course topics (reinforcement learning and control for manipulation), but it also overlaps with my own research projects, one of which involves using differentiable physics simulators to solve robust optimization and control problems. This class project overlaps with my main research in that it uses the same tools and explores related problems, but it adds a focus on contact-rich tasks (which was not previously a part of my main research, but might become one of the example problems I use after seeing these results). Additionally, this project explores the comparison between gradient-based and gradient-free methods, which was not part of my main research project.

As a simple summary of the relationship between this project and my main research: I would likely not have done this project if I had not taken 6.843, but now that I have done this project I think it will integrate well with my other projects (perhaps showing up as an additional example in a future paper or thesis).

References

[1] M. J. Kochnderfer and T. A. Wheeler, *Algorithms for optimization*. The MIT Press, 2019.

[2] I. Mordatch, E. Todorov, and Z. Popović, "Discovery of complex behaviors through contact-invariant optimization," *ACM Trans. Graph.*, vol. 31, no. 4, jul 2012. [Online]. Available: https://doi-org.libproxy.mit.edu/10.1145/2185520.2185539

[3] M. Posa, C. Cantu, and R. Tedrake, "A direct method for trajectory optimization of rigid bodies through contact," *The International Journal of Robotics Research*, vol. 33, no. 1, pp. 69–81, 2014. [Online]. Available: https://doi.org/10.1177/0278364913506757

[4] H. J. T. Suh, T. Pang, and R. Tedrake, "Bundled gradients through contact via randomized smoothing," 2021.

[5] A. Nagabandi, K. Konoglie, S. Levine, and V. Kumar, "Deep Dynamics Models for Learning Dexterous Manipulation," in *Conference on Robot Learning (CoRL)*, 2019.

[6] H. Zhu, A. Gupta, A. Rajeswaran, S. Levine, and V. Kumar, "Dexterous manipulation with deep reinforcement learning: Efficient, general, and low-cost," in *2019 International Conference on Robotics and Automation (ICRA)*, 2019, pp. 3651–3657.

[7] J. Achiam, "Spinning Up in Deep Reinforcement Learning," 2018.

[8] J. Rapin and O. Teytaud, "Nevergrad - A gradient-free optimization platform," https://GitHub.com/FacebookResearch/Nevergrad, 2018.

[9] E. Heiden, D. Millard, E. Coumans, Y. Sheng, and G. S. Sukhatme, "NeuralSim: Augmenting differentiable simulators with neural networks," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2021. [Online]. Available: https://github.com/google-research/tiny-differentiable-simulator

[10] 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 *Advances in Neural Information Processing Systems*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds., vol. 31. Curran Associates, Inc., 2018. [Online]. Available: https://proceedings.neurips.cc/paper/2018/file/842424a1d0595b76ec4fa03c46e8d755-Paper.pdf

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

[12] J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. VanderPlas, S. Wanderman-Milne, and Q. Zhang, "JAX: composable transformations of Python+NumPy programs," 2018. [Online]. Available: http://github.com/google/jax