# Towards a Suite of Smoothing-Based Local Solvers for Contact-Rich Trajectory Optimization

Ajinkya Bhole[1,2], Mohammad Mahmoudi Filabadi[1,2], Guillaume Crevecoeur[1,2], Tom Lefebvre[1,2]
[1]Department of Electromechanical, Systems and Metal Engineering, Ghent University, Ghent, Belgium
[2]Core lab MIRO, Flanders Make, Belgium

***Abstract*—Contact-rich robotic tasks fundamentally challenge trajectory optimization (TO) due to non-smooth dynamics and frequent mode transitions that produce non-informative or discontinuous gradients. Recent smoothing-based methods address this by injecting controlled stochasticity to smooth the optimization landscape, enabling derivative-based algorithms to operate effectively in non-smooth settings. This paper reviews and compares three such methods, namely State-Control-Smoothed DDP (SCS-DDP), Control-Smoothed DDP (CS-DDP), and our recent work Probabilistic DDP (P-DDP), each employing DDP-like backward–forward passes that produce time-varying affine feedback controllers and enable *contact-implicit* planning. While none of these methods is uniformly superior, each offers distinct trade-offs in smoothing scope, sample requirements, and stochasticity scheduling. We present a structured comparison, demonstrate all methods on representative non-smooth robotic tasks involving contact and dry friction, and discuss our vision of developing these approaches into a robust, interchangeable suite of local solvers that helps practitioners identify which method best suits a given problem and can be integrated into broader planning and control architectures for contact-rich robotics. Project video available at: https://youtu.be/fgrfEzwXOFs**

## I. Introduction

As robots transition from controlled settings into real-world environments, they must interact with their surroundings through sustained and purposeful contact. Tasks such as manipulation, tool use, and locomotion require reasoning about intermittent contact, frictional forces, and complex multi-modal dynamics [1]–[3].

Trajectory optimization (TO) is a central tool for generating dynamically feasible motions. Algorithms like Differential Dynamic Programming (DDP) [4] and its first-order variant iLQR exploit local approximations of dynamics and cost to compute optimal trajectories. A key strength is that they naturally produce a *time-varying affine feedback* control law, providing immediate robustness to perturbations and making them attractive as online controllers within MPC schemes [5].

However, non-smooth phenomena are ubiquitous in contact-rich robotics. Impacts, making-and-breaking of contact, and dry friction produce non-informative or discontinuous gradients [6], rendering standard derivative-based TO unreliable. Derivative-free approaches like reinforcement learning (RL) have shown remarkable success in such settings [7], [8], but at the cost of high sample complexity, illustrating the no-free-lunch trade-off: disregarding derivative information comes at the cost of reduced sample efficiency. Recent research attributes RL's effectiveness in non-smooth settings to its intrinsic stochastic exploration, which performs a mechanism akin to *randomized smoothing* (RS): injecting stochasticity averages over and smooths out the non-smoothness inherent in the problem, enabling effective gradient-based updates [9]. This insight has inspired methods that combine the sample efficiency of model-based optimal control with stochastic exploration [3], [10], [11], occupying a middle ground between model-based and learning-based approaches that capitalizes on their combined strengths. Additionally, the growing availability of massively parallel GPU-accelerated physics simulators [12]–[14], which can execute thousands of dynamics rollouts simultaneously, is making the sample requirements of these methods increasingly practical, further motivating their adoption as viable components in modern robotics stacks.

As one might expect, a key consideration when incorporating stochasticity in these methods, is how it is *scheduled* over the optimization iterations: it must be sufficient initially to smooth the problem and encourage exploration, while decaying gradually to converge to an optimal trajectory for the original non-smooth problem. The three smoothing-based TO methods reviewed in this paper: SCS-DDP [3], CS-DDP [10], and P-DDP [11], differ precisely in how they implement this scheduling, as well as in other important design choices: what part of the problem is smoothed (dynamics only vs. the entire cost-to-go), the dimensionality of the sampling space (and hence sample complexity), and whether analytical derivatives of the dynamics are required. A central finding of our structured analysis is that no single method is uniformly superior; rather, each offers its own benefits and limitations along the aforementioned comparison axes. This motivates our vision of developing these approaches into a robust, interchangeable *suite* of local solvers, with clear guidelines on when to use which method, helping the robotics community make informed choices for different problem settings.

Another important observation is that, as local trajectory optimizers, these methods are inherently greedy and can get trapped in local minima when tasks require long-horizon, multi-step reasoning. Overcoming this limitation requires embedding them within hierarchical control architectures alongside global planners. A compelling example of this approach is the recent work of Suh et al. [15], which pairs a global roadmap with a local MPC planner. Their work achieves highly efficient local planning by tightly coupling their core algorithm to custom, differentiable, quasi-static contact models. But this can inherently face a "reality gap" when bridging plans back to true second-order physics, frequently requiring heuristic interventions to prevent failures upon deployment.

In contrast, the smoothing-based methods emphasized in this paper offer an interchangeable spectrum of simulator agnosticism. While sample-efficient variants leverage available analytical derivatives, the fully zero-th order variants treat the physics simulator strictly as a black box. By natively preserving true second-order dynamics across these options, this paradigm empowers practitioners to directly plug-and-play with existing physics engines, enabling contact-rich planning without the engineering burden of differentiable models.

**Algorithm 1** SCS-DDP algorithm [3]

**Input:** TO problem: $C(\xi_{0:T}), f_t(\xi_t)$; initial state $x_0$, input sequence $u_{0:T}$; initial covariance $\Sigma_{\xi\xi,0}$; decay parameters $\beta$; target precision $\alpha^*$

**Output:** $u^*_{0:T-1}$

Initialize $\xi_{0:T}$
**repeat**
  $k^*_{0:T}, K^*_{0:T}$ ▷ *Backward Pass*
  $\xi_{0:T}$ ▷ *Forward Pass*
  $\Sigma_{\xi\xi} \leftarrow \Sigma_{\xi\xi}/\beta$
**until** $|\Delta C| < \alpha^*$

## II. Background and Methods

### A. Trajectory Optimization and the Non-Smooth Challenge

Consider the deterministic trajectory optimization problem:

$$\begin{aligned} \min_{u_{0:T-1}} \quad & C(\xi_{0:T}) := c_T(x_T) + \sum_{t=0}^{T-1} c_t(\xi_t) \\ \text{s.t.} \quad & x_{t+1} = f_t(\xi_t), \quad t = 0, \dots, T-1, \end{aligned} \tag{1}$$

where $\xi_t = (x_t, u_t)$, $x_t \in \mathbb{R}^{n_x}$ is the state, $u_t \in \mathbb{R}^{n_u}$ is the control, and $f_t$, $c_t$, $c_T$ are the dynamics, stage cost, and terminal cost.

DDP iterates between a *backward pass*, which recursively computes a quadratic approximation of the state-action value function $Q_t(\xi_t)$ and derives the optimal policy $\delta u^*_t = k^*_t + K^*_t \delta x_t$ (with $k^*_t = -Q^{-1}_{uu,t} Q_{u,t}$, $K^*_t = -Q^{-1}_{uu,t} Q_{ux,t}$), and a *forward pass* that simulates under the updated policy.

For non-smooth systems, the dynamics Jacobian is null in certain modes (e.g., sticking, no contact), making $Q_u = 0$ and trapping derivative-based methods [9], [10]. Randomized smoothing addresses this: convolving a locally Lipschitz function $g$ with a Gaussian kernel yields a smooth approximation $\hat{g}(x) = \mathbb{E}[g(x + C\epsilon)]$ whose gradient can be estimated in zero-th order form as $\nabla_x \hat{g}(x) = \mathbb{E}[g(x + C\epsilon) \otimes C^{-1}\epsilon]$, requiring only function evaluations [3], [16].

### B. SCS-DDP: Smoothing in State–Control Space [3]

State-Control-Smoothed DDP (SCS-DDP)[1] replaces the dynamics and its derivatives in the DDP backward pass with their smoothed counterparts via zero-th order RS, sampling perturbations $\epsilon \sim \mathcal{N}(0, I)$ in the joint state–control space $\mathbb{R}^{n_x+n_u}$:

$$\hat{f}_t(\xi_t) = \mathbb{E}\big[f_t(\xi_t + C_{\xi\xi}\epsilon)\big], \; \hat{F}_{\xi,t} = \mathbb{E}\big[f_t(\xi_t + C_{\xi\xi}\epsilon) \otimes C^{-1}_{\xi\xi}\epsilon\big] \tag{2}$$

where $C_{\xi\xi}$ is the Cholesky factor of $\Sigma_{\xi\xi}$. Being fully zero-th order, SCS-DDP requires no analytical derivatives from the simulator. The covariance $\Sigma_{\xi\xi}$ is decayed each iteration by a factor $\beta$, following the Robbins–Monro rule [17]. The overall approach is summarized in Alg. 1.

### C. CS-DDP: Smoothing in Control Space Only [10]

Control-Smoothed DDP (CS-DDP)[2], restricts perturbations to the control dimension, $\tilde{u}_t = u_t + C_{uu}\epsilon$ with $\epsilon \in \mathbb{R}^{n_u}$, ensuring only physically reachable states are explored. However, this restriction is both a bane and a boon: while the lower-dimensional sampling space ($n_u$ vs. $n_x + n_u$) provides a significant sample efficiency advantage when $n_x > 0$, restricting the smoothing to only the control space means the method still requires access to the analytical state derivatives $\nabla_x f_t$. CS-DDP follows the same algorithmic structure as SCS-DDP (Alg. 1), but with the covariance $\Sigma_{\xi\xi}$ replaced by $\Sigma_{uu}$.

[1]SCS-DDP was originally proposed as a variant of iLQR termed iMPC [3], as the algorithm incorporated state and input constraints. Without constraints, it reduces to standard iLQR.

[2]CS-DDP was originally referred to as Randomized DDP in [10].

**Algorithm 2** P-DDP algorithm [11]

**Input:** TO problem: $C(\xi_{0:T}), f_t(\xi_t)$; initial state $x_0$, control policy $\pi_{0:T-1}$, initial covariance $\Sigma_{\xi\xi,0}$; risk-sensitivity $\lambda$; target precision $\alpha^*$

**Output:** $u^*_{0:T-1}$

Initialize $\mu_{\xi,0:T}$ and $\Sigma_{\xi\xi,0:T}$
**repeat**
  $k^*_{0:T}, K^*_{0:T}, \Sigma^*_{0:T}$ ▷ *Backward Pass*
  $\mu_{\xi,0:T}, \Sigma_{\xi\xi,0:T}$ ▷ *Forward Pass*
  $\lambda \leftarrow \lambda^*$ ▷ *Optional: risk sensitivity update*
**until** $|\Delta C| < \alpha^*$

### D. P-DDP: Probabilistic DDP [11]

Probabilistic DDP (P-DDP) takes a fundamentally different route. It exploits the insight that for deterministic dynamics, the solution of the TO problem and the Risk-Sensitive Optimal Control (RSOC) problem coincide [18]. Casting RSOC as probabilistic inference on a graphical model, the problem becomes amenable to the Expectation–Maximization (EM) algorithm [19], [20]. A major benefit of this framing over the heuristic scheduling in SCS-DDP and CS-DDP is that stochastic exploration emerges naturally, providing a *principled mechanism* for automatic stochasticity scheduling directly from the EM updates.

The controller is parameterized as a Gaussian policy $\pi_t(u_t|x_t) = \mathcal{N}(\delta u_t;\; k_t + K_t \delta x_t,\; \Sigma_t)$, where the covariance $\Sigma_t$ represents the exploration noise. Algorithmic execution begins with a forward pass that propagates the uncertainty of the prior policy through the dynamics to obtain a joint state–control distribution $p(\xi_t|\pi_{0:t}) \approx \mathcal{N}(\xi_t; \mu_{\xi,t}, \Sigma_{\xi\xi,t})$. This forward propagation yields a nominal trajectory *distribution*. The resulting covariance structure is then used in the backward pass to compute zero-th order smoothed estimates of the state–action ($Q_t$) and value ($V_t$) functions, yielding the EM-based optimal policy updates [11]:

$$\begin{aligned} & K^*_t = \Sigma^*_t(\Sigma^{-1}_t K_t - \lambda \hat{Q}^*_{ux,t}), \quad k^*_t = \Sigma^*_t(\Sigma^{-1}_t k_t - \lambda \hat{Q}^*_{u,t}), \\ & \Sigma^*_t = (\Sigma^{-1}_t + \lambda \hat{Q}^*_{uu,t})^{-1} \end{aligned} \tag{3}$$

where $\lambda > 0$ is the risk-sensitivity parameter. The overall algorithm is outlined in Alg. 2.

This structure affords two key properties: (i) *Global smoothing scope:* Because P-DDP estimates the expected values of the value functions themselves in the backward pass (rather than just the dynamics), it implicitly smooths the entire problem. This allows it to naturally handle non-smooth costs (e.g., $\ell_1$ penalties or constraint indicators) alongside non-smooth dynamics. (ii) *Automatic scheduling:* The stochasticity decay is managed implicitly by the EM update for the policy covariance (3) and the optimal risk sensitivity parameter

TABLE I
COMPARISON OF SMOOTHING-BASED TRAJECTORY OPTIMIZATION METHODS FOR NON-SMOOTH ROBOTIC TASKS.

| | **SCS-DDP** [3] | **CS-DDP** [10] | **P-DDP** [11] |
|---|---|---|---|
| **Framework** | DDP with smoothed dynamics | DDP with smoothed dynamics | Probabilistic optimal control (RSOC + EM), yielding a DDP-like algorithm |
| **What is smoothed** | Dynamics $f_t$ in both state and control space | Dynamics $f_t$ in control space only | Entire problem: value and state-action value functions (implicitly smoothing cost *and* dynamics) |
| **Smoothing dimension** | $n_x + n_u$ | $n_u$ | $n_x + n_u$ |
| **Sample complexity** | High (samples in $\mathbb{R}^{n_x+n_u}$) | Low (samples in $\mathbb{R}^{n_u}$ only) | High (samples in $\mathbb{R}^{n_x+n_u}$) |
| **Requires $\nabla_x f_t$/$\nabla^2_x f_t$?** | No (fully zero-th order) | Yes, for $\nabla_x \hat{f}_t$, $\nabla^2_x \hat{f}_t$, $\nabla_{ux} \hat{f}_t$ | No (fully zero-th order) |
| **Cost gradient/Hessian** | Exact (analytical) | Exact (analytical) | Smoothed (estimated via RS along with value functions) |
| **State feasibility** | May produce infeasible (penetrating) states | Only physically reachable states explored | May produce infeasible (penetrating) states |
| **Non-smooth costs** | Not handled (exact cost derivatives assumed) | Not handled (exact cost derivatives assumed) | Handled (costs are smoothed together with value functions) |
| **Stochasticity scheduling** | Heuristic: with decay factor $\beta$ | Heuristic: with decay factor $\beta$ | Automatic via EM, can also use heuristic rules |

$\lambda^* = (T+1)/\mathbb{E}[C(\xi_{0:T})]$ [11]. In practice, a purely EM-based schedule can be sensitive to arbitrary cost scaling, it can supplemented with heuristic components, such as precision-weighting the prior $\Sigma_t^{-1}$ in (3) or applying multiplicative bounds on $\lambda$.

## III. STRUCTURED COMPARISON

Table I summarizes the key differences among the three methods across multiple axes.

SCS-DDP and P-DDP inject noise in the full state–control space, while CS-DDP restricts to controls. CS-DDP thus requires fewer samples, which is advantageous when $n_x > 0$. CS-DDP explores only feasible states, while SCS-DDP/P-DDP may produce infeasible configurations (the physics engine must therefore be able to handle such cases). Furthermore, CS-DDP needs analytical $\nabla_x f_t$, while SCS-DDP/P-DDP are fully zero-th order. P-DDP uniquely handles non-smooth costs but can introduce noise into cost gradient estimates, making stability of the backward pass more sensitive to sample count. All three share a common DDP structure producing affine feedback controllers deployable in a MPC fashion.

## IV. SIMULATION EXPERIMENTS

We evaluated all three methods on representative non-smooth robotic tasks involving contact interactions and dry friction, demonstrating that smoothing-based approaches enable *contact-implicit* planning: the optimizer discovers contact mode sequences automatically without pre-specified schedules.

**Block pushing (1 pusher):** A pusher must maneuver a rigid block to a target SE(2) pose. The cost used is contact-implicit: a proximity term based on a smooth signed-distance function encouraging the particle to approach the block surface, quadratic penalties driving the block toward its goal pose, and velocity and effort terms regularizing the motion. No pushing strategy, i.e. which face, when, or where to contact, is prescribed. Standard DDP initially drives the pusher toward the block, but because no contact forces exist prior to collision, the gradients vanish ($\partial x^+_{\text{block}}/\partial u = 0$) and it fails to generate the feedforward actions required to push it. In contrast, the smoothing-based methods successfully diffuse these sparse gradients. This allows the optimizer to "feel" the block from afar, organically synthesizing a feasible pushing strategy.

**Block pushing with obstacle (2 pushers):** The challenge scales as two pushers must now coordinate to steer the block around a fixed spherical obstacle, with added clearance penalties to prevent collisions. Once again, standard DDP stalls upon contact. The smoothing-based solvers, however, discover highly sophisticated coordinated maneuvers. Remarkably, they even learn to leverage the obstacle itself to deliberately pivot and redirect the block, demonstrating an emergent form of *extrinsic dexterity* [21].

**Cartpole with friction:** An underactuated cartpole is tasked with swinging up to the inverted equilibrium, fighting against dry Coulomb friction at the joint. Standard DDP successfully drives the cart to the goal but becomes permanently trapped in the non-smooth stiction regime, failing to invert the pole. By effectively injecting noise to explore beyond this local stiction basin, all three smoothing methods cleanly overcome the friction trap and swing the pole to the top.

**Hyperparameter sensitivity:** Additionally, we observed that all smoothing-based methods exhibit notable sensitivity to algorithm hyperparameters, which directly govern the optimizer's exploration-exploitation trade-off. We provide a detailed breakdown of this hyperparameter sensitivity in the Appendix.

## V. VISION: A SUITE OF LOCAL SOLVERS FOR CONTACT-RICH ROBOTICS

The comparative study above suggests that no single smoothing-based method is uniformly superior. Each offers distinct trade-offs in sample efficiency, derivative require-

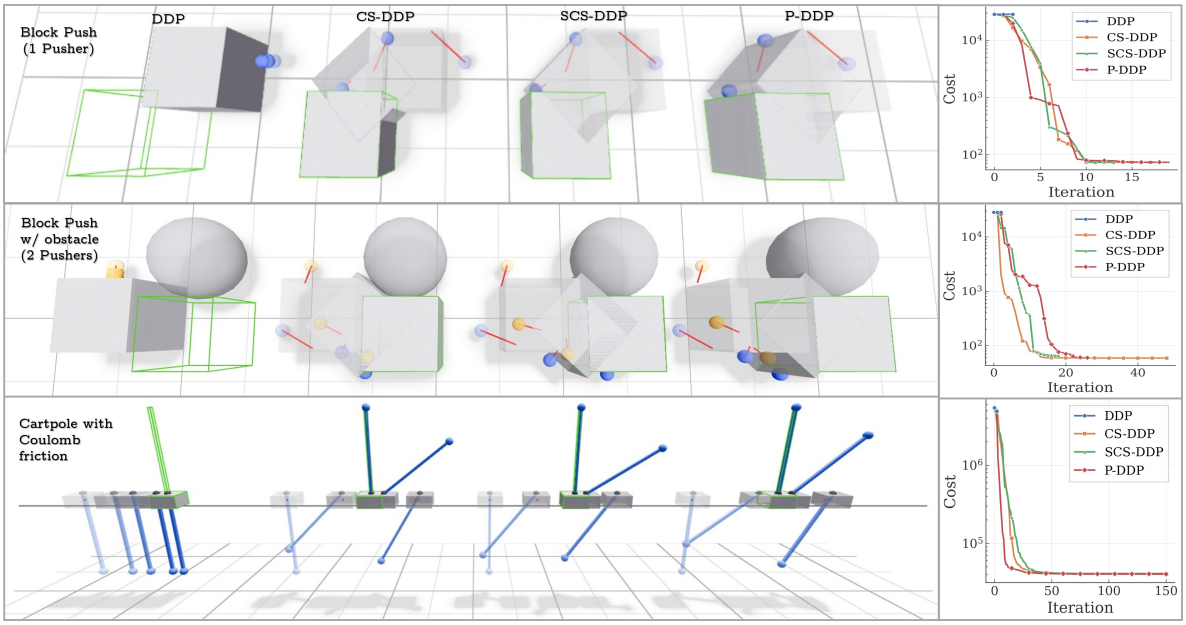


Fig. 1. Trajectory snapshots (columns 1–4: DDP, CS-DDP, SCS-DDP, P-DDP) and cost convergence (column 5) for three experiments.

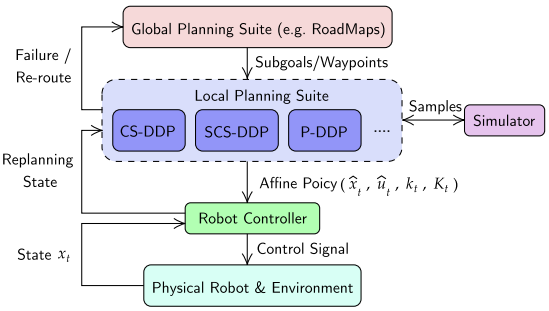


Fig. 2. Envisioned hierarchical control architecture for contact-rich robotics.

ments, smoothing scope, and scheduling mechanisms. This observation motivates our central thesis: rather than seeking a single best method, we envision developing these approaches into a *robust, interchangeable suite* of local solvers for contact-rich trajectory optimization.

The local solvers discussed in this paper are highly effective for contact-rich maneuvers but inherently lack the global reasoning required for long-horizon tasks. Crucially, we emphasize that this suite is not intended as a standalone, monolithic planning package. Instead, it acts specifically as a high-performance local control backend. To successfully navigate complex, long-horizon tasks, it inherently relies on a higher-level global planner, such as a sampling-based roadmap or a kinodynamic search tree, to decompose the combinatorial complexity of the workspace and provide reachable, intermediate subgoals for the local solvers to track. This motivates embedding this suite within a hierarchical control architecture, as shown in Fig. 2.

Translating this conceptual architecture into practical utility requires developing an open-source, user-friendly software API. Unlike rigid frameworks that force a single algorithmic choice, this modular API would allow researchers to seamlessly hot-swap solvers based on their specific constraints. For instance, switching to CS-DDP when analytical derivatives are cheaply available, or reverting to P-DDP when the task involves highly non-smooth cost landscapes. By standardizing the interface between the local solvers and the physics simulator, we aim to drastically reduce the engineering overhead for deploying contact-rich controllers.

## VI. Conclusion

This paper presented a comparative study of three smoothing-based trajectory optimization methods for non-smooth robotic tasks. All methods enable contact-implicit planning and produce affine feedback controllers, but differ in their smoothing scope, sample requirements, derivative dependencies, and stochasticity scheduling mechanisms. Our experiments demonstrate that while each method can discover complex contact strategies autonomously, including emergent extrinsic dexterity, all remain sensitive to hyperparameter tuning, and no single method dominates. SCS-DDP and CS-DDP use fixed heuristic decay; P-DDP offers automatic scheduling through EM but benefits from supplementary rules. A key question is whether *adaptive, problem-aware* strategies can robustify the exploration-to-exploitation transition across diverse tasks, and will form one of the directions of our future work. Furthermore, to systematically evaluate the "no-free-lunch" trade-offs of these local solvers, our future efforts will also focus on establishing a standardized suite of contact-rich benchmarking problems. Due to the inherent stochasticity of these methods, a critical direction for future work is conducting a rigorous, systematic hyperparameter sensitivity analysis. Relying on isolated trials is insufficient; we must establish proper frameworks to analyze and explain how variance in these parameters impacts the exploration-exploitation trade-off across multiple runs. By developing a robust methodology to evaluate this sensitivity, we aim to formulate concrete guidelines that help practitioners select and configure the optimal solver for a given task.

## REFERENCES

[1] I. Mordatch, E. Todorov, and Z. Popović, "Discovery of complex behaviors through contact-invariant optimization," *ACM Trans. Graph.*, vol. 31, no. 4, pp. 43:1–43:8, 2012.

[2] M. Posa, C. Cantu, and R. Tedrake, "A direct method for trajectory optimization of rigid bodies through contact," in *Int. J. Robot. Res.*, vol. 33, no. 1, 2014, pp. 69–81.

[3] T. Pang, H. J. T. Suh, L. Yang, and R. Tedrake, "Global planning for contact-rich manipulation via local smoothing of quasi-dynamic contact models," *IEEE Trans. Robot.*, vol. 39, pp. 4691–4711, 2023.

[4] D. H. Jacobson and D. Q. Mayne, *Differential Dynamic Programming*. New York: American Elsevier Pub. Co., 1970.

[5] J. Z. Zhang, T. A. Howell, Z. Yi, C. Pan, G. Shi, G. Qu, T. Erez, Y. Tassa, and Z. Manchester, "Whole-body model-predictive control of legged robots with MuJoCo," *arXiv preprint arXiv:2503.04613*, 2025.

[6] K. Werling, D. Omens, J. Lee, I. Exarchos, and C. K. Liu, "Fast and feature-complete differentiable physics for articulated rigid bodies with contact," in *Proc. Robotics: Sci. Syst. (RSS)*, 2021.

[7] A. Rajeswaran, V. Kumar, A. Gupta, G. Vezzani, J. Schulman, E. Todorov, and S. Levine, "Learning complex dexterous manipulation with deep reinforcement learning and demonstrations," *arXiv preprint arXiv:1709.10087*, 2017.

[8] J. Hwangbo, J. Lee, A. Dosovitskiy, D. Bellicoso, V. Tsounis, V. Koltun, and M. Hutter, "Learning agile and dynamic motor skills for legged robots," *Sci. Robot.*, vol. 4, p. eaau5872, 2019.

[9] H. J. T. Suh, T. Pang, and R. Tedrake, "Bundled gradients through contact via randomized smoothing," *IEEE Robot. Autom. Lett.*, vol. 7, pp. 4000–4007, 2022.

[10] Q. Le Lidec, F. Schramm, L. Montaut, C. Schmid, I. Laptev, and J. Carpentier, "Leveraging randomized smoothing for optimal control of nonsmooth dynamical systems," *Nonlinear Anal. Hybrid Syst.*, vol. 52, p. 101468, 2024.

[11] M. M. Filabadi, T. Lefebvre, and G. Crevecoeur, "Leveraging probabilistic optimal control for efficient trajectory optimization," *International Journal of Robust and Nonlinear Control*, 2026.

[12] V. Makoviychuk, L. Wawrzyniak, Y. Guo, M. Lu, K. Storey, M. Macklin, D. Hoeller, N. Rudin, A. Allshire, A. Handa, and G. State, "Isaac Gym: High performance GPU-based physics simulation for robot learning," in *Proc. Neural Information Processing Systems (NeurIPS) Datasets and Benchmarks Track*, 2021.

[13] C. D. Freeman, E. Frey, A. Raichuk, S. Girber, I. Mordatch, and O. Bachem, "Brax – a differentiable physics engine for large scale rigid body simulation," *arXiv preprint arXiv:2106.13281*, 2021.

[14] E. Todorov, T. Erez, and Y. Tassa, "MuJoCo: A physics engine for model-based control," *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, pp. 5026–5033, 2012.

[15] H. J. T. Suh, T. Pang, T. Zhao, and R. Tedrake, "Dexterous contact-rich manipulation via the contact trust region," *arXiv preprint arXiv:2505.02291*, 2025.

[16] J. Duchi, M. Jordan, M. Wainwright, and A. Wibisono, "Optimal rates for zero-order convex optimization: The power of two function evaluations," *IEEE Trans. Inf. Theory*, vol. 61, pp. 2788–2806, 2015.

[17] H. Robbins and S. Monro, "A stochastic approximation method," *Ann. Math. Statist.*, pp. 400–407, 1951.

[18] T. Lefebvre, "Probabilistic control and majorisation of optimal control," *Syst. Control Lett.*, vol. 190, p. 105837, 2024.

[19] S. Levine, "Reinforcement learning and control as probabilistic inference: Tutorial and review," *arXiv preprint arXiv:1805.00909*, 2018.

[20] K. C. Rawlik, "On probabilistic inference approaches to stochastic optimal control," Ph.D. dissertation, University of Edinburgh, 2013.

[21] N. C. Dafle, A. Rodriguez, R. Paolini, B. Tang, S. S. Srinivasa, M. Erdmann, M. T. Mason, I. Lundberg, H. Staber, and T. Fuhlbrigge, "Extrinsic dexterity: In-hand manipulation with external forces," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 2014, pp. 1578–1585.


## APPENDIX: HYPERPARAMETER SENSITIVITY

All smoothing-based methods are subject to hyperparameter tuning, which regulates the pace of exploration versus exploitation. In SCS-DDP and CS-DDP, the initial sample covariance is reduced multiplicatively by a heuristic decay factor $\beta > 1$ at each iteration. P-DDP schedules stochasticity automatically through its EM formulation, updating the policy covariance and the risk sensitivity $\lambda$; however, in practice this pure EM update can be augmented with heuristics to prevent premature convergence. A multiplicative schedule $\kappa$ can be used to decay the risk sensitivity, and a precision-weighting factor $\eta$ can relax the prior precision during the backward pass to deliberately encourage robust exploration [11].

We explored this sensitivity by repeating the two-pusher obstacle task under three hyperparameter settings (Fig. 3). The configurations (left to right) are $(\beta,\, \kappa,\, \eta) = (3,\, 4,\, 0.05)$, $(6,\, 5,\, 0.01)$, and $(5,\, 4,\, 0.01)$.

The qualitative convergence behavior is heavily dependent on these settings. In the first setting, CS-DDP converges slowly and stagnates suboptimally, while SCS-DDP plateaus early. In the second setting, despite P-DDP utilizing the exact same parameters as in the successful run shown in Fig. (1), it falls into a local minimum, illustrating algorithm failure strictly due to Monte Carlo sampling variance. Exploring how to robustify this exploration-exploitation handover, perhaps using problem-aware heuristics, remains a critical direction for our future work.

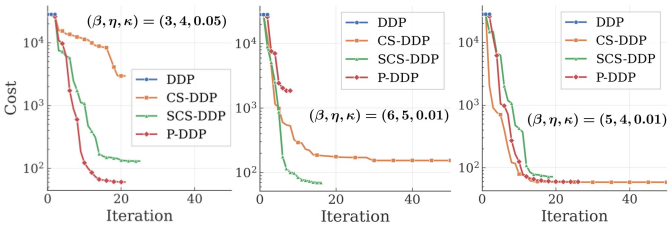


Fig. 3. Cost convergence for the two-pusher obstacle task under three different hyperparameter settings.