An Introduction to Zero-Order Optimization Techniques for Robotics

Armand Jordana, Jianghan Zhang, Joseph Amigo and Ludovic Righetti

***Abstract*— Zero-order optimization techniques are becoming increasingly popular in robotics due to their ability to handle non-differentiable functions and escape local minima. These advantages make them particularly useful for trajectory optimization and policy optimization. In this work, we propose a mathematical tutorial on random search. It offers a simple and unifying perspective for understanding a wide range of algorithms commonly used in robotics. Leveraging this viewpoint, we classify many trajectory optimization methods under a common framework and derive novel competitive RL algorithms.**

I. Introduction

In recent years, zero-order (or derivative-free) optimization techniques [1] have gained a lot of popularity in the robotics community. While zero-order optimization is a well-established field, its widespread deployment in robotics has only been made possible by recent advances in parallel computing and GPU hardware. These improvements have made it possible to deploy sampling-based Model Predictive Control (MPC) on complex robotic systems [2], [3]. In parallel, Reinforcement Learning (RL) has emerged as a powerful tool and has demonstrated state-of-the-art capabilities in locomotion or manipulation [4], [5].

While these approaches may appear unrelated at first, they share a common property: they do not require access to the simulator's gradients. This characteristic allows them to optimize non-smooth objective functions, which typically arise in problems involving contact, such as locomotion or manipulation. Another benefit is that it avoids the tedious development of efficient gradient implementations. Furthermore, while deterministic gradient-based algorithms, such as gradient descent or Newton's method, are prone to getting stuck in local minima, zero-order techniques are mostly stochastic, which can help escape local minima.

Zero-order optimization has a long history [6]. In its most generic form, zero-order optimization performs a random search by generating samples at random and updating an estimate of the optimal solution based on the performance of each sample. In the 1970s, Evolutionary Strategies (ES) gained popularity [7] and later culminated with the CMA-ES algorithm [8]. In TO, a popular algorithm is MPPI [9], which was initially derived using information-theoretic arguments. In RL, Reinforce [10] was one of the first major algorithms proposed for continuous action space problems. These algorithms have since inspired many variations.

Armand Jordana is part of LAAS-CNRS, France.

Jianghan Zhang, Joseph Amigo and Ludovic Righetti are part of the Machines in Motion Laboratory, New York University, USA.

Corresponding author: armand.jordana@laas.fr

While most of these algorithms have strong theoretical foundations, this is not always widely recognized within the robotics community. One reason for this is that these algorithms often lie at the intersection of many fields, such as optimization, statistics, machine learning, and control. By bringing together results from these areas, our goal is both to bring to light the theoretical properties of existing algorithms and to leverage this understanding to design novel algorithms for robotics.

Recently, [11] introduced Gaussian smoothing and understood random search from an optimization perspective. Inspired by this work, we propose a unified perspective to understand the zero-order optimization techniques that are popular in the robotics community. More specifically, we are interested in algorithms that can find solutions to the following problem:

$$\min_{x \in \mathbb{R}^n} f(x). \tag{1}$$

This formulation encompasses a wide range of problems encountered in robotics, such as trajectory optimization (TO) and RL.

In the TO community, several works have studied derivative-free algorithms and established connections among them. For instance, [12] first showed the similarities between MPPI and CMA-ES. Later, [13] understood MPPI as an approximate gradient method, and [3] connected MPPI to diffusion models. Our work builds on these insights and proposes a unifying framework to understand these algorithms. In addition, we shed light on the connection between MPPI and recent results showcasing the benefits of the log-sum-exp transform [14]. Lastly, we benchmark several state-of-the-art approaches on various robotic examples.

In the RL community, [15], [16] first explored the use of random search in the parameter space of policies as an alternative to traditional RL algorithms. In this work, we instead investigate how random search techniques can explain the success of popular RL algorithms. More recently, [17], [18] drew the connection between policy gradient techniques and Nesterov's random search [11]. However, these works used this insight to derive TO algorithms based on random search. In contrast, we study how to leverage this understanding to derive novel RL algorithms. Lastly, we refer the reader to [19], [20] for comprehensive surveys on the connection between RL and ES techniques.

Overall, a broad perspective connecting gradient-free approaches in TO and RL is missing. To bridge this gap, we propose a mathematical introduction to zero-order optimization algorithms used in robotics. In addition, we show

how this unified view allows us to naturally derive novel competitive methods as a byproduct.

II. Background on Gaussian Smoothing

Gaussian smoothing [11] approximates the gradient of a given function f by the gradient of a surrogate function :

$$f_\mu(x) = \mathbb{E}[f(x + \mu\epsilon)], \tag{2}$$

where $\epsilon \sim \mathcal{N}(0, \Sigma)$. This is commonly referred to as randomized smoothing (RS). As μ approaches zero, f_μ tends to f. As a result, for small values of μ, this surrogate only slightly changes the objective function (and therefore the minimum location) but adds smoothness [11]. Importantly, the advantage of this surrogate is that its gradient can be evaluated via an average of function evaluations. More precisely, one can show that:

$$\nabla f_\mu(x) = \mathbb{E}\left[\frac{1}{\mu} f(x + \mu\epsilon)\Sigma^{-1}\epsilon\right]. \tag{3}$$

Hence, the gradient of f_μ can be estimated using only function evaluations of the original function, f. However, this comes at the cost of approximating an expectation with samples. Lastly, since $\mathbb{E}\left[f(x)\Sigma^{-1}\epsilon\right] = 0$, we can write:

$$\nabla f_\mu(x) = \mathbb{E}\left[\frac{f(x + \mu\epsilon) - f(x)}{\mu}\Sigma^{-1}\epsilon\right]. \tag{4}$$

In the general case, the expectation is intractable. The idea is then to approximate the gradient of the original function with a stochastic estimate of the gradients of the surrogate:

$$x \leftarrow x - \frac{f(x + \mu\epsilon) - f(x)}{\mu}\Sigma^{-1}\epsilon. \tag{5}$$

While $\frac{1}{\mu} f(x + \mu\epsilon)\Sigma^{-1}\epsilon$ is a valid gradient estimate, it can be arbitrarily large. In contrast, the update (5) is invariant to constant translations of the function. Intuitively, this reduces the variance of the estimate. In the convex setting, this iterative procedure (Eq. (5)) requires at most n times more iterations than the standard gradient method [11]. In other words, sampling along random directions may match the performance of gradient descent while requiring n times fewer function evaluations.

III. Trajectory Optimization

In this section, we show how random search allows us to understand derivative-free optimization algorithms that are commonly used in robotics. In practice, TO aims to solve

$$\min_{(u_0, u_1, \dots u_{T-1})} \sum_{t=0}^{T-1} c_t(x_t, u_t) + c_T(x_T), \tag{6}$$
$$\text{such that } x_0 = x, \text{ and } x_{t+1} = f_{\text{dyn}}(x_t, u_t).$$

Here x denotes the state, u the control variable, $(c_t)_t$ the running cost functions, c_T the terminal cost function, T the time horizon, and f_{dyn} the dynamics. Note that we abuse the notation by using x to denote the state instead of the optimization variable. Importantly, the constraints are implicit; the optimization is performed only with respect to the control variables. Therefore, this problem is unconstrained and is referred to as the single shooting approach. Ultimately, it can be written in the form of Problem (1). The dimension of the search space is $n = Tn_u$, where n_u denotes the dimension of the control inputs.

A. The log-sum-exp transform: MPPI

Model Predictive Path Integral (MPPI) [9] is a derivative-free TO method initially derived from an information-theoretic perspective. MPPI iteratively samples around the current guess and updates it using a weighted exponential average. More specifically, given a current guess x, MPPI samples K variables x_k following independent Gaussian distributions centered in x (i.e. $x_k \sim \mathcal{N}(x, \Sigma)$). Then, it performs an update according to the following rule:

$$x \leftarrow \sum_{k=1}^{K} w_k x_k, \text{where } w_k = \frac{\exp(-\frac{1}{\lambda} f(x_k))}{\sum_{j=1}^{K} \exp(-\frac{1}{\lambda} f(x_j))}. \tag{7}$$

These weights, w_k, are called Exponential Average weights. Let's see how this update rule can be interpreted as a variation of Gaussian smoothing. Instead of the surrogate function defined in Equation (2), let's consider the (continuous) log-sum-exp transform function.

$$f_{\mu,\lambda}(x) = -\lambda \log\left(\mathbb{E}\left[\exp\left(-\frac{1}{\lambda} f(x + \mu\epsilon)\right)\right]\right), \tag{8}$$

where $\epsilon \sim \mathcal{N}(0, \Sigma)$ and where $\lambda > 0$ is a scalar called the temperature. As μ approaches zero, we recover f. Therefore, the gradients of this surrogate can be used to approximate the gradients of the original function, f. Furthermore, as $\lambda \to \infty, f_{\mu,\lambda} \to f_\mu$ [14]. Consequently, this surrogate can be seen as a generalization of Gaussian smoothing. The gradient of the surrogate reads:

$$\nabla f_{\mu,\lambda}(x) = \frac{-\lambda \mathbb{E}\left[\exp\left(-\frac{1}{\lambda} f(x + \mu\epsilon)\right)\Sigma^{-1}\epsilon\right]}{\mu \mathbb{E}\left[\exp\left(-\frac{1}{\lambda} f(x + \mu\epsilon)\right)\right]}. \tag{9}$$

In practice, both expectations can be approximated with K samples $\epsilon_{1:K}$ following the distribution $\mathcal{N}(0, \Sigma)$. This leads to the following gradient estimate:

$$g = \frac{-\lambda}{\mu} \frac{\sum_{k=1}^{K} \exp\left(-\frac{1}{\lambda} f(x + \mu\epsilon_k)\right)\Sigma^{-1}\epsilon_k}{\sum_{j=1}^{K} \exp\left(-\frac{1}{\lambda} f(x + \mu\epsilon_j)\right)}. \tag{10}$$

Similarly to the randomized smoothing case, this update is invariant to additive translation of the function f. Let's consider $\mu = 1$. We show that MPPI follows a natural gradient step [21], i.e. $x \leftarrow x - \alpha F^{-1} g$, where α is the step size, and where F is the Fisher information matrix, which is equal to the inverse of the covariance matrix for Gaussian distributions. The idea of the natural gradient is to build an update step that is invariant to changes of variables. If $\alpha = \frac{1}{\lambda}$, we have

$$x - \alpha F^{-1} g = \sum_{i=1}^{K} w_k (x + \epsilon_k). \tag{11}$$

As $(x+\epsilon_k) \sim \mathcal{N}(x, \Sigma)$, we recover the MPPI update. More detailed derivations are provided in [22]. Furthermore, if $\Sigma = \sigma^2 I$, then $\frac{\sigma^2}{\lambda} \leq \frac{1}{L}$, where L is the Lipschitz function of the surrogate function [14]. Therefore, the step size chosen by MPPI can be interpreted as a conservative estimate of the standard optimal step size from convex optimization [23].

B. Covariance Matrix Adaptation (CMA)

So far, we have kept the covariance matrix Σ fixed across various iterations. However, in practice, it is not necessarily clear how to choose this parameter according to the problem. In this section, we show how to derive a Covariance Matrix Adaptation (CMA) scheme. Interestingly, the optimization of the smooth surrogate function can be interpreted as an optimization in the space of Gaussian probability distributions. Indeed, optimizing Equation (2) can be interpreted as a search over the mean of a Gaussian distribution. More specifically, we can write:

$$f_\mu(x) = \mathbb{E}[f(x+\mu u)] = \mathbb{E}_{z\sim\mathcal{N}(x,\Sigma)}\left[f(z)\right]. \tag{12}$$

Therefore, Gaussian smoothing can be seen as the search for a Gaussian distribution that minimizes the expectation of f under that distribution. This distribution can be interpreted as the belief of where the global minimum of the function f might be. A natural generalization is to not only optimize the mean but also the covariance. More specifically, one can apply gradient descent to the following problem:

$$\min_{\theta=(x,\Sigma)} J(\theta), \text{ where } J(\theta) = \mathbb{E}_{z\sim\mathcal{N}(x,\Sigma)}\left[f(z)\right]. \tag{13}$$

$J(\theta)$ is minimal when the distribution $\mathcal{N}(x, \Sigma)$ concentrates around the minima of f [24]. [25] shows that the natural gradient update, $\Delta\theta = F(\theta)^{-1}\nabla J(\theta)$, can be written:

$$\begin{aligned}\Delta\Sigma &= \mathbb{E}_{z\sim\mathcal{N}(x,\Sigma)}\left[f(z)((z-x)(z-x)^\top - \Sigma)\right],\\ \Delta x &= \mathbb{E}_{z\sim\mathcal{N}(x,\Sigma)}\left[f(z)(z-x)\right].\end{aligned} \tag{14}$$

Similarly to the randomized smoothing case, this update step averages quantities that rely solely on function evaluations of the original function. Following the natural gradient is especially relevant in this setting, as it is crucial that the update does not depend on the choice of parameterization of the Gaussian distribution. In practice, this update step can be approximated by samples $x_k \sim \mathcal{N}(x, \Sigma)$. Denoting $w_k = f(x_k)$, the approximate update with a step size α reads:

$$\begin{aligned}\Sigma &\leftarrow (1-\alpha\sum_{k=1}^{K} w_k)\Sigma + \alpha\sum_{k=1}^{K} w_k(x_k-x)(x_k-x)^\top,\\ x &\leftarrow (1-\alpha\sum_{k=1}^{K} w_k)x + \alpha\sum_{k=1}^{K} w_k x_k.\end{aligned} \tag{15}$$

This allows us to recover the CMA scheme. As with Gaussian smoothing, a constant can be subtracted from f without changing the expectation in Equation (14). Therefore, another choice of weights could be $w_k = f(x_k) - f(x)$, as this renders the update invariant to translations of f. One could go further and design an algorithm invariant to any increasing transformation of the objective by replacing $f(x_k)$ with arbitrary weights w_k sorted so that their order matches that of the function values $f(x_k)$. As pointed out by [24]–[26], this recovers the CMA-ES algorithm (without evolution path) [8]. An important point that we overlooked is whether the covariance update rule in Equation (15) maintains the matrix positive definite. If $0 \leq \eta < 1$, $w_i \geq 0$, and the weights sum to one, then the covariance matrix is always positive definite, provided that we start with a positive definite matrix [25]. However, in practice, there is no reason for the function f to be such that the weights satisfy this property. One approach could be to use the log-sum-exp transformation employed for MPPI. Indeed, this transformation naturally maintains the weights positive and ensures that they sum to one. We omit the derivations as it is straightforward to show that applying the natural gradient to Equation (8) yields the update rule from Equation (15) with the Exponential Average weights (Equation (7)). We refer to this algorithm as MPPI-CMA.

C. Numerical experiments

We introduce a benchmark based on a balancing task with the G1 humanoid. All results are averaged over six seeds. We compare the performance of Predictive Sampling, randomized smoothing, MPPI, and MPPI-CMA. All algorithms use 2048 samples per iteration. For MPPI and MPPI-CMA, we use a temperature of $\lambda = 0.1$. We find that MPPI-CMA performs significantly better when using a separate step size for the mean update and the covariance update. Figure 1 shows the cost evolution across iterations for each test problem. The performance of randomized smoothing varies across test problems and is highly sensitive to the choice of step size. As expected, MPPI-CMA consistently outperforms MPPI. Interestingly, Predictive Sampling performs well despite its simplicity. [22] provides more experimental results.

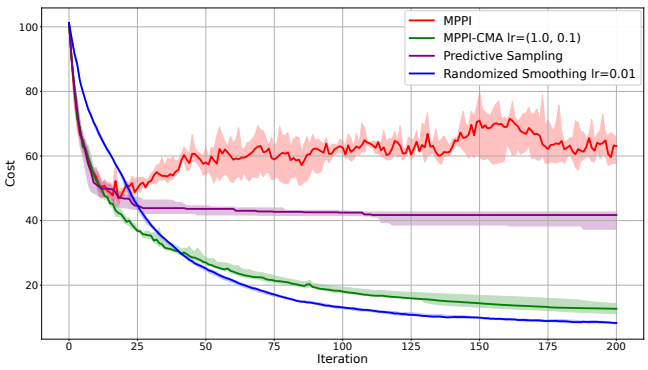

Fig. 1: Cost according to the number of iterations for different TO test problems. The solid lines represent the median taken over six seeds.

IV. Policy Optimization

One may ask how this novel understanding of gradient-free techniques can impact RL. In this section, we show how this

unified treatment can be used to derive novel algorithms. In the deterministic setting, RL aims to find a policy. This can be written as the maximization over θ of :

$$F(\theta) = \mathbb{E}\left[J(\theta, s)\right], \tag{16}$$

where the expectation is taken over initial states s and where $J(\theta, s)$ is the cost of the trajectory with initial state s and with the controller π_θ:

$$J(\theta, s) = \sum_{k=0}^{\infty} \gamma^k r(s_k, a_k), \tag{17}$$
$$\text{s.t. } s_0 = s, s_{k+1} = f_{\text{dyn}}(s_k, a_k) \text{ and } a_k = \pi_\theta(s_k).$$

θ denotes the parameters of the policy (e.g. the weights of a neural network). r is the reward, a is the control action. We do not consider stochastic dynamics for simplicity, as most robotic models are deterministic (e.g. locomotion or manipulation). However, the ideas presented in this section should extend to the stochastic dynamics case. Because the policy is deterministic, RL algorithms aiming to solve formulation (16) are called Deterministic Policy Gradient (DPG) algorithms [27]. Most RL algorithms rely on stochastic policies, therefore DPG algorithms are often introduced as a special case. While stochastic policies can be relevant in the context of games or partially observable settings, most robotic applications, such as locomotion or manipulation, can be solved with deterministic policies. In fact, in practice, many works that rely on RL algorithms derived with stochastic policies generally deploy a deterministic policy on the robot. Therefore, we chose to focus on the deterministic case and refer the reader to [22] for a detailed discussion on the stochastic case.

The DPG theorem [27] states that :

$$\nabla_\theta F(\theta) = \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k \partial_\theta \pi_\theta(s_k) \partial_a Q(s_k, a_k)\right]. \tag{18}$$

Unfortunately, this formula cannot be directly implemented in an algorithm because of the presence of the Q-function. Indeed, the Q-function cannot be computed analytically. One solution is to estimate it with function approximation at each gradient descent step. Algorithms relying on this mechanism are called actor-critics. The actor refers to the policy, while the critic refers to the Q-function. In practice, the Q-function is learned in a self-supervised way via the Bellman equation. There exist many techniques to estimate the critic efficiently between gradient updates of the actor; we refer the reader to [10] for a comprehensive treatment of the subject.

DDPG [28] originally proposed to learn the Q-function with a neural network and directly differentiate it to obtain $\partial_a Q(s_k, a_k)$. This idea led to state-of-the-art RL algorithms such as TD3 [29]. However, there are no guarantees that the derivatives of the learned Q-function are correct. In light of the success of randomized smoothing, a natural idea is to use Gaussian smoothing to estimate the gradient of the Q-function and to use the following estimate :

$$(Q(s_k, a_k + \epsilon_k) - Q(s_k, a_k))\Sigma^{-1}\epsilon_k. \tag{19}$$

This provides us with a simple variation of actor-critics based on the DPG theorem. An interesting variation is to use the log-sum-exp transform and employ exponential average weights in the actor update. We deliberately omit the rollout logic, as well as the interplay between the actor update and the critic update. This is because our focus is solely on modifying the actor's gradient update in DPG algorithms. We do not aim to provide a novel actor-critic logic.

Our experiments show that this simple change can lead to competitive algorithms. We investigate the benefits of estimating the gradient of the Q-function in DPG algorithms such as DDPG and TD3. Our implementation relies on CleanRL [30] and only modifies the actor's update rule. Specifically, we investigate two smoothing approaches, the default randomized smoothing (RS) and the one relying on the log-sum-exp (LSE) transform. Consequently, we benchmark DDPG and TD3 against their smoothed counterparts: RS-DDPG, LSE-DDPG, RS-TD3, and LSE-TD3. The performances are evaluated in seven MuJoCo environments. Five runs are performed for each test problem. Figure 2 presents the normalized score across all runs. The results indicate that both RS-DDPG and LSE-DDPG significantly outperform DDPG, which demonstrates the benefits of smoothing. For TD3, the improvements are not as clear. This could be explained by the fact that TD3 is already a very strong algorithm and that the margin for improvement on those benchmarks is limited. Additionally, our modifications were restricted to the actor's update; achieving state-of-the-art performance would most likely require extensive hyperparameter tuning of the actor-critic mechanism. Nevertheless, the results show that deterministic policy gradient algorithms can benefit from smoothing. More details are provided in [22].

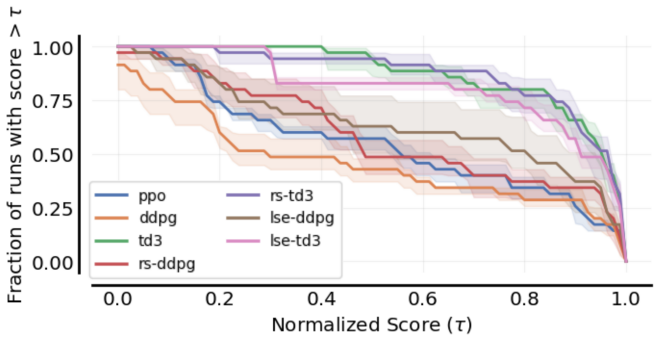

Fig. 2: Performance of each RL algorithm.

V. Conclusion

In this work, we have demonstrated how random search provides a unifying perspective on zero-order algorithms commonly used in robotics. We also discussed theoretical concepts that help explain why sampling-based zero-order techniques can escape local minima. Leveraging this understanding, we proposed novel and competitive RL algorithms. For an extended version of this work, we refer the reader to [22].

REFERENCES

[1] A. R. Conn, K. Scheinberg, and L. N. Vicente, *Introduction to derivative-free optimization*. SIAM, 2009.

[2] A. H. Li, P. Culbertson, V. Kurtz, and A. D. Ames, "Drop: Dexterous reorientation via online planning," *arXiv preprint arXiv:2409.14562*, 2024.

[3] H. Xue, C. Pan, Z. Yi, G. Qu, and G. Shi, "Full-order sampling-based MPC for torque-level locomotion control via diffusion-style annealing," *arXiv preprint arXiv:2409.15610*, 2024.

[4] A. Handa, A. Allshire, V. Makoviychuk, A. Petrenko, R. Singh, J. Liu, D. Makoviichuk, K. Van Wyk, A. Zhurkevich, B. Sundaralingam, and Y. Narang, "Dextreme: Transfer of agile in-hand manipulation from simulation to reality," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 5977–5984.

[5] D. Hoeller, N. Rudin, D. Sako, and M. Hutter, "Anymal parkour: Learning agile navigation for quadrupedal robots," *Science Robotics*, vol. 9, no. 88, p. eadi7566, 2024.

[6] J. Matyas, "Random optimization," *Automation and Remote control*, vol. 26, no. 2, pp. 246–253, 1965.

[7] H.-P. P. Schwefel, *Evolution and optimum seeking: the sixth generation*. John Wiley & Sons, Inc., 1993.

[8] N. Hansen and A. Ostermeier, "Completely derandomized self-adaptation in evolution strategies," *Evolutionary computation*, vol. 9, no. 2, pp. 159–195, 2001.

[9] G. Williams, P. Drews, B. Goldfain, J. M. Rehg, and E. A. Theodorou, "Aggressive driving with model predictive path integral control," in *2016 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2016, pp. 1433–1440.

[10] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: A Bradford Book, 2018.

[11] Y. Nesterov and V. Spokoiny, "Random gradient-free minimization of convex functions," *Foundations of Computational Mathematics*, vol. 17, no. 2, pp. 527–566, 2017.

[12] F. Stulp and O. Sigaud, "Path integral policy improvement with covariance matrix adaptation," in *Proceedings of the 29th International Conference on Machine Learning (ICML-12)*, ser. ICML '12, J. Langford and J. Pineau, Eds. New York, NY, USA: Omnipress, July 2012, pp. 281–288.

[13] N. Wagener, C. Cheng, J. Sacks, and B. Boots, "An online learning approach to model predictive control," in *Robotics: Science and Systems XV, University of Freiburg, Freiburg im Breisgau, Germany, June 22-26, 2019*, A. Bicchi, H. Kress-Gazit, and S. Hutchinson, Eds., 2019. [Online]. Available: https://doi.org/10.15607/RSS.2019.XV.033

[14] K. Scaman, L. Dos Santos, M. Barlier, and I. Colin, "A simple and efficient smoothing method for faster optimization and local exploration," *Advances in Neural Information Processing Systems*, vol. 33, pp. 6503–6513, 2020.

[15] T. Salimans, J. Ho, X. Chen, S. Sidor, and I. Sutskever, "Evolution strategies as a scalable alternative to reinforcement learning," *arXiv preprint arXiv:1703.03864*, 2017.

[16] H. Mania, A. Guy, and B. Recht, "Simple random search of static linear policies is competitive for reinforcement learning," *Advances in neural information processing systems*, vol. 31, 2018.

[17] H. J. T. Suh, T. Pang, and R. Tedrake, "Bundled gradients through contact via randomized smoothing," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 4000–4007, 2022.

[18] 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 Analysis: Hybrid Systems*, vol. 52, p. 101468, 2024.

[19] O. Sigaud and F. Stulp, "Policy search in continuous action domains: an overview," *Neural Networks*, vol. 113, pp. 28–40, 2019.

[20] O. Sigaud, "Combining evolution and deep reinforcement learning for policy search: A survey," *ACM Transactions on Evolutionary Learning*, vol. 3, no. 3, pp. 1–20, 2023.

[21] S.-I. Amari, "Natural gradient works efficiently in learning," *Neural computation*, vol. 10, no. 2, pp. 251–276, 1998.

[22] A. Jordana, J. Zhang, J. Amigo, and L. Righetti, "An introduction to zero-order optimization techniques for robotics," *arXiv preprint arXiv:2506.22087*, 2025.

[23] Y. Nesterov, *Lectures on convex optimization*. Springer, 2018, vol. 137.

[24] Y. Ollivier, L. Arnold, A. Auger, and N. Hansen, "Information-geometric optimization algorithms: A unifying picture via invariance principles," *Journal of Machine Learning Research*, vol. 18, no. 18, pp. 1–65, 2017.

[25] Y. Akimoto, Y. Nagata, I. Ono, and S. Kobayashi, "Bidirectional relation between cma evolution strategies and natural evolution strategies," in *Parallel Problem Solving from Nature, PPSN XI: 11th International Conference, Kraków, Poland, September 11-15, 2010, Proceedings, Part I 11*. Springer, 2010, pp. 154–163.

[26] D. Wierstra, T. Schaul, T. Glasmachers, Y. Sun, J. Peters, and J. Schmidhuber, "Natural evolution strategies," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 949–980, 2014.

[27] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic policy gradient algorithms," in *International conference on machine learning*. Pmlr, 2014, pp. 387–395.

[28] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[29] S. Fujimoto, H. Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," in *International conference on machine learning*. PMLR, 2018, pp. 1587–1596.

[30] S. Huang, R. F. J. Dossa, C. Ye, J. Braga, D. Chakraborty, K. Mehta, and J. G. Araújo, "Cleanrl: High-quality single-file implementations of deep reinforcement learning algorithms," *Journal of Machine Learning Research*, vol. 23, no. 274, pp. 1–18, 2022. [Online]. Available: http://jmlr.org/papers/v23/21-1342.html