

# A Controlled Perturbation Algorithm for Saddle Point Escape of Generic Non-convex Optimization Problems

(Algorithm Description – Version 1.0)

Ka-Hei Cheng (鄭家禧)\*

May 2026

## Abstract

We introduce the Controlled Perturbation Algorithm (CPA) for escaping saddle points in generic non-convex optimization problems. The key idea is to use two adaptive perturbations per coordinate, evaluate their directional derivatives, and deterministically select a descent direction – all without computing second or higher order derivatives. We also define the Non-Descent Direction Approximation (NDDA) index as a cheap heuristic indicator of proximity to a local minimum.

**This note is a preliminary algorithmic description intended to establish priority.** No experimental validation is included here. A subsequent extended version will provide empirical results, code, and comparisons with existing methods. The algorithm is presented as a heuristic tool; rigorous convergence guarantees are left for future work.

## Note to the reader

The author’s background is in physics, not in formal optimization theory. Consequently, this work prioritizes a clear physical picture – treating the loss landscape as an energy landscape and using controlled perturbations to “feel” the local curvature – over mathematical rigor. The algorithm and the NDDA index are presented as *heuristic tools*. Formal guarantees, convergence rates, and detailed comparisons with state-of-the-art optimizers are left for future work.

## Version and priority statement

This document (Version 1.0) is deposited on Zenodo on May 1, 2026 to establish priority for the algorithm described herein. A full version with experimental validation, code, and multi-seed statistical analysis will be released as Version 2.0. This work is licensed under CC BY 4.0.

---

\*Email: khcheng920911@gmail.com

# 1 Background and motivation

Non-convex optimization lies at the heart of modern machine learning, signal processing, robotics, and many scientific computing applications. The loss landscapes in these domains are typically rugged, containing a multitude of critical points – local minima, local maxima, and saddle points. While local minima are desirable, saddle points can severely impede the convergence of first-order methods such as gradient descent (GD). At a strict saddle point, the gradient is zero but the Hessian possesses at least one negative eigenvalue, making it an unstable equilibrium. Near such points, GD stalls because it lacks directional information to descend further.

Classical strategies to escape saddle points include adding random noise (perturbed gradient descent, or PGD), using momentum, or leveraging second-order curvature information (e.g., trust-region methods, negative curvature search). Among these, PGD has gained popularity due to its simplicity and theoretical guarantees: by occasionally injecting isotropic Gaussian noise, it escapes strict saddles with high probability and converges to second-order stationary points. However, PGD has notable limitations:

- **Inefficiency of random kicks:** The perturbation direction is purely random. Many kicks may be wasted before a descent direction is accidentally found.
- **Difficulty with non-strict saddles:** When the Hessian is positive semi-definite (some eigenvalues zero), PGD often fails because there is no negative curvature to exploit; the algorithm may wander on a plateau for an extremely long time.
- **Lack of a cheap convergence indicator:** Standard stopping criteria rely on gradient norm, which does not distinguish between saddles and minima. Reliable detection of minima often requires Hessian information, which is computationally prohibitive in high dimensions.

These shortcomings motivate the need for a deterministic, curvature-agnostic method that can (i) reliably identify a descent direction near saddles without relying on random chance, (ii) handle both strict and non-strict saddles, and (iii) provide an inexpensive, real-time measure of how close the iterate is to a local minimum.

## 2 Heuristic physical picture

Suppose we would like to minimize a function  $E(\theta)$ , with dimension of  $\theta$  being  $d$ . Suppose the system is at  $\theta = \theta_0$ . The idea of the Controlled Perturbation Algorithm is, for any  $i$ -th coordinate, an  $\varepsilon_i$  is chosen randomly. Gradient would then be computed at perturbed point  $\theta_0 + \varepsilon$ , whose sign on  $i$ -th coordinate would reflect whether the function is ascent or descent direction along the move  $\varepsilon_i$  in this direction. If it is descent, this direction would be our desired move. On the other hand, if the direction is ascent, we may consider its opposite direction. If this opposite direction is a descent one, this would be the desired direction. But if this direction is also ascent, it would be an indicator that the system may have a high chance in local minimum along this  $i$ -th coordinate. Therefore, we do not change the coordinate in this direction.

Making use of the above idea, we propose the Non-Descent Direction Approximation (NDDA) Index on the ratio of non-descent directions in both forward and backward

probes. In local minimum, this index would be expected to be close to 1.

A rigorous saddle point escape rate for the Controlled Perturbation Algorithm (CPA) is beyond the scope of this work. Nevertheless, we offer a heuristic argument for why its escape efficiency should be at least competitive with perturbed gradient descent (PGD) [4]. In PGD, a random perturbation is added uniformly in all directions to help the iterates leave saddle points. While this suffices to guarantee polynomial-time escape, a large fraction of the random directions may not correspond to descent directions in the local geometry, potentially requiring many attempts. In contrast, CPA uses an additional gradient evaluation to bias the perturbation toward directions of steepest descent. Therefore, we expect that CPA wastes fewer iterations on non-descent perturbations, and under similar smoothness conditions its escape efficiency should be at least as good as that of PGD. A formal verification of this intuition is an interesting direction for future work.

Regarding the application of CPA and the NDDA index, we currently propose using them only once the system reaches a stationary point where the gradient vanishes. Specifically, CPA is applied to escape the stationary point, with the NDDA index serving as an indicator of whether a descent direction is present; after escape, the optimizer can switch to a traditional algorithm such as Adam or standard gradient descent.

When the steepest descent direction does not align with the coordinate axes, our per-coordinate perturbation can still approximate off-axis directions because the perturbation vector is built by independently choosing the sign for each coordinate based on the local gradient information.

Moreover, even if the selected perturbation deviates significantly from the steepest descent direction, as long as it is a descent direction the gradient norm typically increases as the iterate moves away from the flat region, making the subsequent switch to a conventional optimizer increasingly efficient.

We advise restricting CPA to zero-gradient regions at this stage because its convergence properties away from stationary points remain uncharacterized. The primary purpose of CPA is to escape saddle points; preliminary explorations (not included here) suggest that CPA may also be effective as a general optimizer in non-zero gradient regions, a possibility we plan to investigate in future work.

### 3 Controlled Perturbation Algorithm (CPA)

The following pseudocode describes the algorithm. Note that the computational cost is  $O(d)$  per iteration, where  $d$  is the number of parameters (degrees of freedom).

---

**Algorithm 1** Controlled Perturbation Algorithm (CPA)

---

**Require:** Objective function  $E(\theta)$ ,  $\theta \in \mathbb{R}^d$ ; initial parameters  $\theta_0$ ; amplification factor  $\alpha \in [1.1, 2.0]$  (default 1.3); noise scale  $\sigma$  (or per-coordinate  $\sigma_i$ ); max iterations  $K$ .

**Ensure:**  $\theta_{\text{final}}$

1: **for**  $t = 0, \dots, K - 1$  **do**

2:   Sample  $\varepsilon_i \sim \mathcal{N}(0, \text{diag}(\sigma_i^2))$  for each coordinate  $i$ .

3:   Compute  $I_{1,i} = \varepsilon_i \left. \frac{\partial E}{\partial \theta_i} \right|_{\theta_0 + \varepsilon}$  for every  $i$ .

4:   Obtain  $\varepsilon'_i$ :

$$\varepsilon'_i = \begin{cases} \alpha \varepsilon_i & \text{if } I_{1,i} < 0 \\ -\varepsilon_i & \text{if } I_{1,i} > 0 \end{cases}$$

5:   Compute the second index at  $\theta_0 + \varepsilon'$ :

$$I_{2,i} = \varepsilon'_i \left. \frac{\partial E}{\partial \theta_i} \right|_{\theta_0 + \varepsilon'}$$

6:   **if**  $I_{2,i} < 0$  **then**

7:      $\theta_{0,i} \leftarrow \theta_{0,i} + \varepsilon'_i$

▷ Descent found along  $\varepsilon'$

8:   **else if**  $I_{1,i} < 0$  and  $I_{2,i} > 0$  **then**

9:      $\theta_{0,i} \leftarrow \theta_{0,i} + \varepsilon_i$

▷ Descent found only along original  $\varepsilon$

10:   **else if**  $I_{1,i} > 0$  and  $I_{2,i} > 0$  **then**

11:      $\theta_{0,i} \leftarrow \theta_{0,i}$

▷ No descent direction in this coordinate

12:   **end if**

13: **end for**

14: **return**  $\theta_{\text{final}}$

---

*Remark:* The case  $I_{1,i} = 0$  or  $I_{2,i} = 0$  is extremely unlikely with random perturbations and continuous gradients; in practice we treat zero as non-descent (no update).

## 4 Non-Descent Direction Approximation (NDDA) Index

The NDDA index is a heuristic indicator of how many coordinates behave like they are at a local minimum. It is computed by the same two perturbations but without performing any parameter update.

---

**Algorithm 2** NDDA Index

---

**Require:** Objective function  $E(\theta)$ ,  $\theta \in \mathbb{R}^d$ ; current parameters  $\theta_0$ ; amplification factor  $\alpha$ ; noise scale  $\sigma$ .

**Ensure:** NDDA  $\in [0, 1]$

- 1: Sample  $\varepsilon_i \sim \mathcal{N}(0, \text{diag}(\sigma_i^2))$ .
  - 2: Compute  $I_{1,i} = \varepsilon_i \frac{\partial E}{\partial \theta_i} \Big|_{\theta_0 + \varepsilon}$ .
  - 3: Obtain  $\varepsilon'_i$  as in CPA.
  - 4: Compute  $I_{2,i} = \varepsilon'_i \frac{\partial E}{\partial \theta_i} \Big|_{\theta_0 + \varepsilon'}$ .
  - 5: NDDA =  $\frac{\#\{i | I_{1,i} \geq 0 \text{ and } I_{2,i} \geq 0\}}{d}$
  - 6: **return** NDDA
- 

When NDDA  $\approx 1$ , the algorithm suggests that for most coordinates neither perturbation direction leads to a descent – this is consistent with being near a local minimum (or a very flat region). Conversely, low NDDA values indicate that many coordinates still have a detectable descent direction, which typically occurs near saddle points or on slopes.

## Remark on perturbation magnitude choices

1. Diminishing parameters: start with a relatively large  $\sigma$  to escape saddles quickly, then decrease it gradually to fine-tune near a minimum.
2. Data-driven scaling: for neural networks, one can pass a batch of data through the model, compute the mean and standard deviation of each layer’s output, and set  $\sigma_i$  proportional to those statistics to keep perturbations in a natural range.

## Acknowledgments

The author thanks the open-source community for providing tools that made this work possible. No external funding was received.

## License

This document is licensed under Creative Commons Attribution 4.0 International (CC BY 4.0). You are free to share and adapt the material, provided you give appropriate credit.

## References

- [1] Dauphin, Y. N., Pascanu, R., Gulcehre, C., Cho, K., Ganguli, S., & Bengio, Y. (2014). Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. *Advances in Neural Information Processing Systems*, **27**.
- [2] Katende, R., & Kasumba, H. (2024). Curvature-Adaptive Perturbation and Subspace Descent for Robust Saddle Point Escape in High-Dimensional Optimization. *arXiv preprint arXiv:2409.12604*.

- [3] Liu, J., & Yuan, Y. (2024). Almost sure convergence rates analysis and saddle avoidance of stochastic gradient methods. *Journal of Machine Learning Research*, **25**(271), 1-40.
- [4] Jin, C., Ge, R., Netrapalli, P., Kakade, S. M., & Jordan, M. I. (2017). How to escape saddle points efficiently. In *International conference on machine learning* (pp. 1724-1732). PMLR.