An Embarrassingly Simple Way to Optimize Orthogonal Matrices at Scale

Adrián Javaloy [1] **Antonio Vergari** [1]

Abstract

Orthogonality constraints are ubiquitous in robust and probabilistic machine learning. Unfortunately, current optimizers are computationally expensive and do not scale to problems with hundreds or thousands of constraints. One notable exception is the Landing algorithm (Ablin et al., 2024) which, however comes at the expense of temporarily relaxing orthogonality. In this work, we revisit and improve on the ideas behind Landing, enabling the inclusion of modern adaptive optimizers while ensuring that orthogonal constraints are effectively met. Remarkably, these improvements come at little to no cost, and reduce the number of required hyperparemeters. Our algorithm POGO is fast and GPU-friendly, *consisting of only* 5 *matrix products*, and in practice maintains orthogonality at all times. On several challenging benchmarks, POGO greatly outperforms recent optimizers and shows it can optimize problems with thousands of orthogonal matrices in minutes while alternatives would take hours. As such, POGO sets a milestone to finally exploit orthogonality constraints in ML at scale. A public PyTorch implementation of POGO is available at github.com/adrianjav/pogo.

1. Introduction

Orthogonality constraints are common in classic machine learning (ML) tasks such as independent and principal component analysis (Bishop, 2006; Hyvärinen et al., 2001). This is also the case in modern ML, where orthogonality plays a crucial role in ensuring the training stability of large recurrent networks (Arjovsky et al., 2016; Kiani et al., 2022), reducing gradient conflict in multitask learning (Javaloy & Valera, 2022), or fostering filter diversity in convolutional neural networks (CNNs) (Wang et al., 2020), vision transformers (ViTs) (Fei et al., 2022) and enabling efficient squared probabilistic circuits (Loconte et al., 2026). While methods such as Riemannian gradient descent (RGD) (Absil et al., 2008) are staples for the classic ML tasks mentioned before, they struggle on modern problems that involve not just one orthogonal matrix, but several (and potentially large) matrices—from dozens to thousands—that interact between each other during training (Li et al., 2019; Fei et al., 2022; Loconte et al., 2026), see Fig. 1.

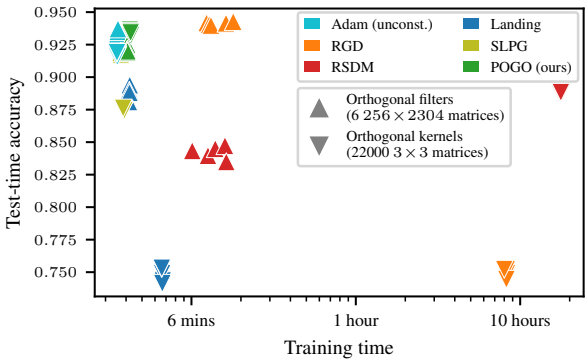

Figure 1. **POGO optimizes thousands of orthogonal matrices orders of magnitude faster than retraction methods while achieving performance comparable to unconstrained optimizers**, as shown in this CIFAR-10 (Krizhevsky et al., 2009) classification problem with a tailored CNN (Jordan, 2024) parameterized with orthogonal filters or kernels. While RSDM (Han et al., 2025) takes **17 hours** to train on average, **POGO trains in 3 minutes**.

This begs the question: How should we design an optimizer for orthogonality constraints, from here on *orthoptimizer* for short, to successfully handle modern large-scale ML problems? We argue that such an orthoptimizer should be: **D1)** ***feasible***, ensuring that the optimized matrices are orthogonal, up to numerical precision; **D2)** ***scalable***, being able to optimize a large number of matrices with minimal overhead with respect to unconstrained deep learning (DL) optimizers; and **D3)** ***competitive***, achieving the best downstream performance possible within the given constraints. Clearly, obtaining all desiderata **D1-3** is challenging, as optimizing for one might trade-off the others, e.g. to enhance scalability (**D2**), one often has to relax orthogonality (**D1**) with the promise to recover it "at the end" of optimization.

One remarkable example of the above is *Landing* (Ablin & Peyré, 2022) and its variants (Ablin et al., 2024; Vary

[1]Institute for Machine Learning, University of Edinburgh, GB. Correspondence to: Adrián Javaloy <ajavaloy@ed.ac.uk>, Antonio Vergari <avergari@ed.ac.uk>.

Proceedings of the 43^{rd} International Conference on Machine Learning, Seoul, South Korea. PMLR 306, 2026.

et al., 2024; Song et al., 2025; Loconte et al., 2026) which, instead of relying on costly retractions in a classic Riemannian fashion, use gradient descent over a modified objective (the *landing field*), requiring only cheap matrix multiplications (**D2**). As a result, however, Landing makes some compromises. First, orthogonality constraints are relaxed during optimization (**D1**), allowing matrices to temporarily violate them until they eventually *land* back to the space of orthogonal matrices. Yet, as we show in §5.2, if several matrices are simultaneously optimized it is uncertain *when* this landing will happen. Second, Landing methods rely entirely on SGD-like updates and therefore miss on techniques like adaptive gradients and momentum that modern optimizers such as Adam (Kingma & Ba, 2015) or Muon (Jordan et al., 2024) bring to DL optimization (**D3**).

In fact, when we measure the raw downstream performance (e.g., classification accuracy) of models with orthogonality constraints trained with Landing—but also with other SoTA orthoptimizers such as RSDM (Han et al., 2025)—they consistently lag behind unconstrained models trained with unconstrained but adaptive optimizers, as shown in Fig. 1. *This highlights an existing gap between theory and practice*: On the one hand, orthogonality should benefit model robustness and stability, as argued above, but on the other hand this promise is never truly met as training constrained models is harder, slower and yields subpar performance.

Contributions. In this paper, we propose an orthoptimizer that substantially reduces this gap, as it proves to be as competitive as Adam on challenging DL benchmarks (**D3**) with thousands of orthogonal matrices, while being orders-of-magnitude faster than retraction methods (**D2**) and closely respecting orthogonality constraints (**D1**), as shown in Fig. 6. Our ***Proximal One-step Geometric Orthoptimizer (POGO)*** extends Landing in several ways. First, POGO can wrap unconstrained optimizers and leverage "optimization tricks" like adaptive momentum (§3.1). Similar to Landing, POGO requires only matrix multiplications and therefore scales gracefully on the GPU. In contrast to Landing, however, POGO is always within an epsilon of the orthogonal manifold under mild conditions, as it can compute exactly the optimal step size to land back on the manifold (§3.2). Furthermore, we prove that under well-behaved gradient norms we can approximate said step size by a constant value (§3.3). Finally, we empirically show (§5) that POGO yields SoTA downstream performance, while being orders-of-magnitude faster than retraction methods and closely respecting orthogonality constraints in a number of benchmarks.

2. Optimization on the Stiefel Manifold

In this work, we look at the following problem:

$$\underset{\boldsymbol{X}_1,\dots,\boldsymbol{X}_L}{\text{minimize}}\ \mathcal{L}(\boldsymbol{X}_1,\dots,\boldsymbol{X}_L) \quad \text{s.t. } \boldsymbol{X}_i\boldsymbol{X}_i^\top = \mathbf{I}_{p_i} \text{ for } i=1,2,\dots,L \tag{1}$$

where $\mathcal{L}$ is any given differentiable function and each of the L matrices $\boldsymbol{X}_i \in \mathbb{R}^{p_i\times n_i}$ is a wide matrix ($p_i \le n_i$) taking values on the real numbers.[1] In practice, $\boldsymbol{X}_i$ can be the attention matrix of a transformer (Fei et al., 2022), the weights of a normalizing flow (Goliński et al., 2019) or recurrent network (Arjovsky et al., 2016), or the filters of a CNN (Wang et al., 2020). As a result, L can go up to hundreds if we, e.g., consider orthogonal filters on a ResNet-110 (Bansal et al., 2018) or even several thousands if one uses orthogonal kernels (Ozay & Okatani, 2016).

A common way to solve Eq. 1 is to optimize over a Riemannian manifold (Absil et al., 2008). In particular, we denote by $\mathrm{St}(p,n)$ the manifold of *row-orthogonal* matrices, with $p \le n$, known as the Stiefel manifold (Stiefel, 1935),

$$\mathrm{St}(p,n) := \left\{\boldsymbol{X} \in \mathbb{R}^{p\times n} \mid \boldsymbol{X}\boldsymbol{X}^\top = \boldsymbol{I}_p\right\}, \tag{2}$$

and consider here for simplicity the Euclidean metric on the Stiefel manifold, i.e., the metric induced by the Frobenius inner product in the ambient space,

$$\langle \boldsymbol{X}, \boldsymbol{Y}\rangle := \mathrm{Tr}(\boldsymbol{Y}^\top\boldsymbol{X}) \quad \text{for } \boldsymbol{X},\boldsymbol{Y} \in \mathbb{R}^{p\times n}. \tag{3}$$

Given $\boldsymbol{X} \in \mathrm{St}(p,n)$, we can define its tangent space, $\mathcal{T}_{\boldsymbol{X}}$, which represents a hyperplane tangent to $\boldsymbol{X}$ where the (Riemannian) gradients w.r.t. $\boldsymbol{X}$ reside. Furthermore, it can be shown that a matrix $\boldsymbol{A} \in \mathcal{T}_{\boldsymbol{X}}$ if and only if it can be written as $\boldsymbol{A} = \boldsymbol{X}\boldsymbol{S}$ where $\boldsymbol{S}$ is a skew-symmetric matrix (Edelman et al., 1998; Ablin et al., 2024).

As described by Ablin & Peyré (2022), we can always compute the Riemannian gradient $\mathrm{grad}(f(\boldsymbol{X}))$ from the Euclidean one as follows: **i)** map $\nabla f(\boldsymbol{X})$ to its relative gradient (Cardoso & Laheld, 2002) by $\boldsymbol{S} := \mathrm{Skew}(\boldsymbol{X}^\top\nabla f(\boldsymbol{X}))$, where $\mathrm{Skew}(\boldsymbol{A}) = 1/2(\boldsymbol{A}-\boldsymbol{A}^\top)$ is the skew operator; and **ii)** compute the Riemannian gradient by left-multiplying $\boldsymbol{S}$ with $\boldsymbol{X}$, i.e., $\mathrm{grad}(f(\boldsymbol{X})) = \boldsymbol{X}\boldsymbol{S} \in \mathcal{T}_{\boldsymbol{X}}$.

Finally, the exponential function $\mathrm{Exp}_{\boldsymbol{X}} : \mathcal{T}_{\boldsymbol{X}} \to \mathrm{St}(p,n)$ provides a way to project $\boldsymbol{A} \in \mathcal{T}_{\boldsymbol{X}}$ back to the manifold. With these tools, we can then define *Riemannian gradient descent* (RGD) (Absil et al., 2008) with learning rate $\eta > 0$ as an iterative algorithm updating $\boldsymbol{X}$ as:

$$\boldsymbol{X}_{t+1} = \mathrm{Exp}_{\boldsymbol{X}_t}(-\eta\,\mathrm{grad}(f(\boldsymbol{X}_t))). \tag{4}$$

Unfortunately, computing the exponential function is expensive and numerically unstable (Arioli et al., 1996; Moler & Loan, 2006). In practice, the exponential function is approximated with *retractions*, i.e., cheaper functions that are locally equivalent to it. In the case of the Stiefel manifold, typical retractions are the QR, Cayley or polar retractions, see for example (Absil et al., 2008, Ex. 4.1.2).

[1] We assume the reals for simplicity, but all derivations can be easily extended to other fields like the complex numbers.

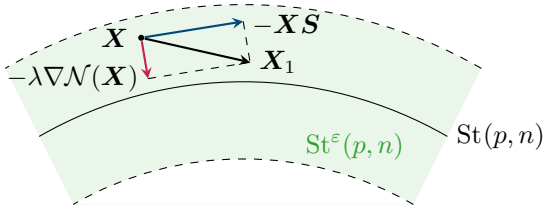

Figure 2. Illustration of the landing algorithm, adapted from (Ablin et al., 2024). Landing combines two orthogonal gradients at each iteration and adapts the learning rate η to keep $\boldsymbol{X}_1$ within ε-distance from the Stiefel manifold.

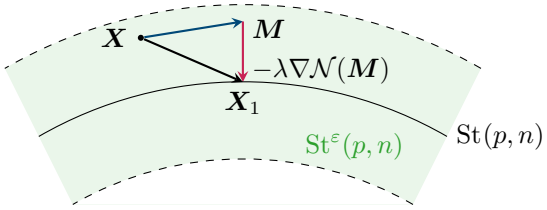

Figure 3. Illustration of the POGO algorithm, see §3. Computing the distance w.r.t. the intermediate point $\boldsymbol{M}$, POGO can calculate the exact λ to stay within the manifold.

However, all of these retractions are either numerically unstable (e.g., the Cayley map involves matrix inversions), or rely on iterative processes (e.g., QR or SVD) that also require GPU-CPU communication (Dongarra et al., 2018). Next, we focus on a more scalable alternative to Riemannian methods: the Landing algorithm.

2.1. The Landing Algorithm

Recently, Ablin & Peyré (2022) introduced Landing as an alternative approach to solve Eq. 1. Instead of using retractions, Landing defines a vector field that pulls iterates to the manifold, letting them escape and eventually *land* back, hence the name. Namely, Landing initializes $\boldsymbol{X}_0 \in \mathrm{St}(p,n)$ and produces the following sequence of iterates:

$$\boldsymbol{X}_{t+1} \coloneqq \boldsymbol{X}_t - \eta\Lambda(\boldsymbol{X}_t)\,, \tag{5}$$

where Λ represents the *landing field*, defined as

$$\Lambda(\boldsymbol{X}) \coloneqq \mathrm{grad}(f(\boldsymbol{X})) + \lambda\nabla\mathcal{N}(\boldsymbol{X})\,, \tag{6}$$

where $\nabla\mathcal{N}(\boldsymbol{X}) = (\boldsymbol{X}\boldsymbol{X}^\top - \mathbf{I}_p)\boldsymbol{X}$ is the gradient of the squared manifold distance, $\mathcal{N}(\boldsymbol{X}) \coloneqq \frac{1}{4}\|\boldsymbol{X}\boldsymbol{X}^\top - \mathbf{I}_p\|^2$. Crucially, both terms in Eq. 6 are *orthogonal* to each other, as depicted in Fig. 2, and thus Landing can be interpreted as seamlessly following a loss-informed direction (Riemannian gradient) and a manifold-attractive one (normal gradient).

As a result, Landing is cheap and GPU-friendly, satisfying **D2**, as it *only requires matrix multiplications*. Ablin et al. (2024) also proved local convergence within the manifold for Landing, conditionally on a bound of the step size η which ensures that no iteration is farther than ε-distance from the manifold. Note that ε needs to be provided as a hyperparameter which defaults to 0.5 otherwise.

Unfortunately, despite promises of an eventual landing, it is uncertain in practice *when exactly* this will occur, as it depends on the training dynamics and the hyperparameters provided to the algorithm (see, e.g., Fig. 5). Similarly, at each iteration the learning rate is computed based on the current point and gradient (Ablin et al., 2024, Lem. 3), and the following *fixed* hyperparameters: **i)** suggested learning rate, η_0; **ii)** manifold attraction strength, λ; and **iii)** maximum manifold distance, ε. This extra complexity results in a trade-off w.r.t. the gained speed-up, hinders the principled use of typical machine learning techniques such as early stopping (Morgan & Bourlard, 1989) and can compromise downstream performance (**D3**). As we show next, this does *not* need to be the case.

3. Proximal One-step Geometric Optimization

In this section, we dissect the update rule of Landing (Eq. 6) and reveal that it already contains the ingredients for an orthoptimizer that is more feasible (**D1**) and competitive (**D3**) while still being scalable (**D2**), leading to our ***Proximal One-step Geometric Orthoptimizer (POGO)***. We provide all proofs for this section in App. A.

3.1. Adopting Modern Optimizers

We argue that one of the reasons holding Landing from being competitive (**D3**) with respect to unconstrained optimizers is its reliance on SGD-like updates, as we can easily observe by unrolling Eqs. 5 and 6:

$$\boldsymbol{X}_{t+1} = \boldsymbol{X}_t - \eta\,\mathrm{grad}(f(\boldsymbol{X}_t)) - \eta\lambda\nabla\mathcal{N}(\boldsymbol{X}_t)\,. \tag{7}$$

However, note that everything introduced in §2.1 as well as the theory provided by Ablin et al. (2024) holds *as long as $\boldsymbol{S}$ is skew-symmetric*, and thus we can safely replace $\nabla f(\boldsymbol{X})$ in $\boldsymbol{S}$ by the output of an unconstrained base optimizer (BO), that is, by $\boldsymbol{G} \coloneqq \mathrm{BO}(\nabla f(\boldsymbol{X}))$. Now, since $\boldsymbol{X}\boldsymbol{S} = \boldsymbol{X}\,\mathrm{Skew}(\boldsymbol{X}^\top\boldsymbol{G})\boldsymbol{X}$ lies in $\mathcal{T}_{\boldsymbol{X}}$ for any $\boldsymbol{G}$, it is worth asking what properties a base optimizer must have to make $\boldsymbol{X}\boldsymbol{S}$ a sensible direction to follow. In this work, we argue that it should be *linear* in the following sense:

Definition 1. An optimizer is linear if its output $\boldsymbol{G}$ is linear w.r.t. its input $\nabla f(\boldsymbol{X})$, up to a possibly $\nabla f(\boldsymbol{X})$-dependent scalar factor, i.e., $\boldsymbol{G} \propto \boldsymbol{A}\nabla f(\boldsymbol{X})$ for a matrix $\boldsymbol{A}$.

Then, since the relative gradient is linear w.r.t. $\boldsymbol{G}$, applying a linear BO becomes an equivariant operation, i.e.,

$$\boldsymbol{S}_{\mathrm{BO}} = \mathrm{Skew}(\boldsymbol{X}^\top\,\mathrm{BO}(\nabla f(\boldsymbol{X}))) \tag{8}$$

$$\propto \mathrm{BO}(\mathrm{Skew}(\boldsymbol{X}^\top\nabla f(\boldsymbol{X})) = \mathrm{BO}(\boldsymbol{S}_{\nabla f(\boldsymbol{X})})\,. \tag{9}$$

In other words, *linear optimizers preserve their semantics*. If we have a linear optimizer, it is equivalent applying it in the Euclidean or tangent space, up to scaling. The vanilla SGD used in Landing, $\boldsymbol{G} = \nabla f(\boldsymbol{X})$, is trivially linear, unlike the popular Adam optimizer, due to its element-wise gradient normalization (Kingma & Ba, 2015). In contrast, Vector Adam (VAdam) (Ling et al., 2022) satisfies Def. 1 as it replaces Adam's normalization with a vector-wise one, see App. B. As we show in §5.3, linear base optimizers that are adaptive, such as VAdam, provide a needed competitive edge in complex optimization landscapes.

3.2. Leap, Land, Repeat

Another aspect to improve in Landing is feasibility (**D1**), as it is only asymptotically guaranteed. Here, we question whether we can land after each iteration or, in other words: *If we were to follow Eq. 7, can we compute the optimal value of λ that puts us back on the manifold?*

Intermediate step. Unfortunately, it seems really difficult to answer the question above if we use Eq. 7. Instead, we propose to reformulate the landing update and follow instead the normal direction of the *intermediate update*:

$$\boldsymbol{M}_t \coloneqq \boldsymbol{X}_t - \eta \boldsymbol{X}_t \boldsymbol{S}\,, \tag{10}$$

$$\boldsymbol{X}_{t+1} \coloneqq \boldsymbol{M}_t - \lambda \nabla \mathcal{N}(\boldsymbol{M}_t)\,. \tag{11}$$

As depicted in Fig. 3, $\boldsymbol{M}_t$ is the intermediate update result of following $\boldsymbol{X}_t\boldsymbol{S}$ in tangent space. This reformulation has two immediate benefits: **i)** we have disentangled the two steps sizes η and λ, as both were entangled in Eq. 7; and **ii)** we can more easily compute the effect that following the normal direction has on $\boldsymbol{X}_{t+1}$, as we show next.

Landing polynomial. Let $P(\lambda)$ be the squared distance of $\boldsymbol{X}_{t+1}$ after following Eqs. 10 and 11, $P(\lambda) \coloneqq 4\mathcal{N}(\boldsymbol{X}_{t+1})$. Then, $P(\lambda)$ turns out to have a nice symbolic form:

Lemma 3.1. *Let $P(\lambda)$ be as defined above, then $P(\lambda)$ is a quartic polynomial w.r.t. λ. Specifically,*

$$\begin{aligned} P(\lambda) = \operatorname{Tr}(\boldsymbol{E}^\top \boldsymbol{E})\lambda^4 + 2\operatorname{Tr}(\boldsymbol{D}^\top \boldsymbol{E})\lambda^3 + \\ + [\operatorname{Tr}(\boldsymbol{D}^\top \boldsymbol{D}) + 2\operatorname{Tr}(\boldsymbol{E}^\top \boldsymbol{D})]\lambda^2 \\ + \operatorname{Tr}(\boldsymbol{C}^\top \boldsymbol{D})\lambda + \operatorname{Tr}(\boldsymbol{C}^\top \boldsymbol{C})\,, \end{aligned}$$

where $\boldsymbol{C} \coloneqq \boldsymbol{M}_t\boldsymbol{M}_t^\top - \boldsymbol{I}$, $\boldsymbol{E} \coloneqq \nabla\mathcal{N}(\boldsymbol{M}_t)\nabla\mathcal{N}(\boldsymbol{M}_t)^\top$ and $\boldsymbol{D} \coloneqq \boldsymbol{M}_t\nabla\mathcal{N}(\boldsymbol{M}_t)^\top + \nabla\mathcal{N}(\boldsymbol{M}_t)\boldsymbol{M}_t^\top$.

We name $P(\lambda)$ the *landing polynomial* and, since it is a quartic polynomial, the problem $\min_\lambda P(\lambda) = 0$ has a *known closed-form solution* using radicals. Furthermore, each coefficient in $P(\lambda)$ can be computed in $\mathcal{O}(p^2 n)$. Note that all these operations are efficiently handled by GPUs.

Choosing a step size. One remaining question is how to pick λ out of the four roots of $P(\lambda)$. If $\boldsymbol{X}$ took values in an algebraically-closed field (e.g., the complex, $\mathbb{C}$), the answer would be simple: the one with the *smallest norm*, minimizing the distance from $\boldsymbol{M}$. In the case of real values, we opt for the *closest real value to any of the roots*, i.e., $\arg\min_{i\in\{1,\dots,4\}} \min_{\lambda_i\in\mathbb{R}} (\lambda_i - \bar{\lambda}_i)^2$ where $\bar{\lambda}_i$ is one root in the algebraic closure of the field. Conveniently, this equals taking the real part of the root with the least imaginary part.

3.3. A Surprising Approximation

While we can efficiently solve the landing polynomial $P(\lambda)$, this still incurs some additional overhead. In this section, we investigate whether there exists a good approximation for λ that is cheaper to compute. To this end, we start by assuming that $\boldsymbol{X}$ lies on the manifold, in which case we can find a simple bound of how far $\boldsymbol{M}$ is from $\operatorname{St}(p, n)$:

Proposition 3.2. *Let $\boldsymbol{X} \in \operatorname{St}(p, n)$ and $\boldsymbol{M} = \boldsymbol{X} - \eta\boldsymbol{X}\boldsymbol{S}$ with $\boldsymbol{S}$ skew-symmetric, then*

$$\|\boldsymbol{M}\boldsymbol{M}^\top - \boldsymbol{I}\| \leq \eta^2 \|\boldsymbol{S}^2\|\,. \tag{12}$$

Remarkably, the bound above depends *exclusively* on the learning rate η and the relative gradient, $\boldsymbol{S}$. Now, since $\boldsymbol{X} \in \operatorname{St}(p, n)$ and $\boldsymbol{S}$ is the skew-symmetric part of $\boldsymbol{X}^\top\boldsymbol{G}$, it is clear that $\|\boldsymbol{S}\|$ is upper-bounded by $\|\boldsymbol{G}\|$. Therefore, we make the following mild assumption:

Assumption 1. $\|\boldsymbol{G}\|$ is upper bounded by L, i.e., $\|\boldsymbol{G}\| \leq L$.

Note that the assumption above bounds the norm of $\boldsymbol{G}$, i.e., *the output of the base optimizer*, and not the Euclidean gradient, $\nabla f(\boldsymbol{X})$. Now, if we define $\xi \coloneqq \eta L$, then Ass. 1 implies that $\eta^2\|\boldsymbol{S}^2\| \leq \xi^2$ (see Prop. A.4). As it turns out, if we keep $\xi < 1$ by e.g. setting the learning rate appropriately, then $\lambda = 1/2$ becomes a great approximation:

Proposition 3.3. *Let $\boldsymbol{X} \in \operatorname{St}(p, n)$ and assume that $\xi < 1$, then we have that $P(1/2) = \mathrm{o}(\xi^7)$.*

This result is quite remarkable: If we keep $\|\boldsymbol{G}\|$ under control, then we can fix λ to $1/2$ and stay within a $\mathrm{o}(\xi^{7/2})$-radius from the manifold. We now move to the general case, where we assume that $\boldsymbol{X}$ is at most ε-far from the manifold, and show that the landing polynomial of $\boldsymbol{X}$ can be bound as a function of its projection to the manifold:

Theorem 3.4. *Let $\boldsymbol{X} \in \mathbb{R}^{p\times n}$ such that $\|\boldsymbol{X}\boldsymbol{X}^\top - \boldsymbol{I}\| \leq \varepsilon$ and further assume that $\xi < 1$. Then*

$$P(\lambda) \leq 2P_{\boldsymbol{Y}}(\lambda) + 2[2 + 2\sqrt{P_{\boldsymbol{Y}}(\lambda)} + Q(\lambda, \varepsilon)]^2 Q(\lambda, \varepsilon)^2\,,$$

where $P_{\boldsymbol{Y}}(\lambda)$ is the landing polynomial of $\boldsymbol{Y}$, the projection of $\boldsymbol{X}$ onto the Stiefel manifold, and where

$$Q(\lambda, \varepsilon) \coloneqq (24\lambda + 3)\varepsilon + \mathrm{o}(\varepsilon) \quad \text{as } \varepsilon \to 0\,. \tag{13}$$

Thus, we can bound the landing polynomial (i.e., the squared distance of $\boldsymbol{X}_{t+1}$ with a step size of λ, see Eq. 11) as a function of only λ and the distance of $\boldsymbol{X}_t$. If we now start reasoning from $\boldsymbol{X}_0 \in \mathrm{St}(p,n)$, we know through Prop. 3.3 that one iteration of Eqs. 10 and 11 with $\lambda = 1/2$ puts $\boldsymbol{X}_1$ at most $\mathrm{o}(\xi^{7/2})$-far from the manifold. Then, we can bound the distance of $\boldsymbol{X}_2$ using Thm. 3.4 with $\varepsilon = \mathrm{o}(\xi^{7/2})$ and $\lambda = 1/2$. Finally, if we continue with this inductive process, we get to the proof of the following result:

Theorem 3.5. *If $\boldsymbol{X}_0 \in \mathrm{St}(p,n)$, $\xi \coloneqq \eta L < 1$ and $\lambda = 1/2$, then for every $\boldsymbol{X}_t$ produced by Eqs. 10 and 11 it holds that*

$$P(1/2) = \|\boldsymbol{X}_t\boldsymbol{X}_t^\top - \boldsymbol{I}\|^2 = \mathrm{o}(\xi^7)\,. \tag{14}$$

In other words, if we ensure that $\xi < 1$ by a combination of **i)** decreasing η and **ii)** using base optimizers with gradient normalization, e.g. VAdam (Ling et al., 2022), then Thm. 3.5 ensures that all iterations $\boldsymbol{X}_t$ from Eqs. 10 and 11 stay close to the manifold if we simply set $\lambda = 1/2$.

Intuition. We note that by substituting $\lambda = 1/2$ in Eq. 11 we obtain $\boldsymbol{X}_{t+1} = (\frac{3}{2}\boldsymbol{I} - \frac{1}{2}\boldsymbol{M}\boldsymbol{M}^\top)\boldsymbol{M}$, which equals the last step of SLPG (Liu et al., 2024), as discussed in App. C. We can leverage this coincidence to shade some light on why $\lambda = 1/2$ is such a good approximation. Namely, Liu et al. (2024) proposed to use the expression above as an approximation to a polar retraction, $(\boldsymbol{M}\boldsymbol{M}^\top)^{-1/2}\boldsymbol{M}$, since $\frac{3}{2} - \frac{1}{2}z$ is a first-order Taylor expansion of $z^{-1/2}$ at 1. As such, we can interpret Eq. 10 ($\boldsymbol{M}_t$) as a gradient step in $\mathcal{T}_{\boldsymbol{X}}$, and Eq. 11 ($\boldsymbol{X}_{t+1}$) as an approximated polar retraction. This intuition further supports Prop. 3.3 and Thm. 3.5.

Convergence. Given the connection of POGO with Landing and the intuition above, the convergence of POGO to a stationary point where $\|\boldsymbol{S}\| = \|\nabla\mathcal{N}(\boldsymbol{X})\| = 0$ could be shown in two ways. First, we can leverage Landing convergence results by reinterpreting Eqs. 10 and 11 as Landing with alternating step sizes (such that $\eta\lambda = 0$ and $\eta + \lambda \neq 0$). Second, we can also interpret POGO for the case $\lambda = 1/2$ and $\boldsymbol{G} = \nabla f(\boldsymbol{X})$ as an $\mathrm{o}(\xi^{7/2})$-approximation of RGD with polar retractions, whose convergence is well-understood, e.g. in Boumal (2023, Cor. 4.9).

3.4. Summary and Practical Considerations

To recapitulate, we provide in Alg. 1 a description of what constitutes POGO, which extends the ideas behind Landing such that, at every step, it approximately finds a point on the manifold proximal to the intermediate update.

Computational cost. As described in Alg. 1, POGO only consists of additions and multiplications, and is therefore GPU-friendly. Namely, POGO needs *5 matrix multiplications* if we set $\lambda = 1/2$, $\mathcal{O}(p^2n)$, and an additional $\mathcal{O}(p^4n)$ otherwise. While POGO requires one more matrix multiplication than Landing, it also removes the need to compute a "step-size safeguard" to keep $\boldsymbol{X}_t$ ε-close to the manifold, and reduces hyperparameters to a minimum: base optimizer, learning rate, and whether to fix or compute λ. Moreover, POGO enjoys three more subtle advantages over retraction-based methods: **i)** since it uses only matrix sums and products, it is numerically stable; **ii)** it benefits from speed-ups in matrix-multiplication primitives such as the reduction of mantissa bits (Henry et al., 2019); and, most remarkably, **iii)** it circumvents the use of *batch SVD*, which is notoriously difficult to parellize on GPUs (Abdelfattah & Fasi, 2026), explaining the significant speed-ups observed in Fig. 1.

Algorithm 1 Proximal One-step Geometric Orthoptimizer.

Input: Input matrix $\boldsymbol{X}$, Euclidean gradient $\nabla f(\boldsymbol{X})$
Parameters: Learning rate η, find root flag, base optimizer BO.

```
let G = BO(∇f(X))                          ▷ SGD, VAdam, etc.
let G = X Skew(X^⊤ G)                      ▷ Riemannian gradient
let M = X − ηG                             ▷ Intermediate step
if find root then
    let λ ← solve_λ P(λ) = 0               ▷ Quartic polynomial
else
    let λ = 1/2                            ▷ λ = 1/2
return M + λ(I − MM^⊤)M
```

Other manifolds. One natural question is whether POGO can be extended to manifolds other than the Stiefel manifold. As we demonstrate in §5.3, this is the case for the complex Stiefel manifold, where $\boldsymbol{X} \in \mathbb{C}^{p\times n}$ and transposes become adjoint matrices. In addition, $\mathrm{St}(n,n)$ and $\mathrm{St}(n-1,n)$ are diffeomorphic to the orthogonal and special orthogonal groups, respectively. Other cases require careful consideration for the manifold field and potential functions, as also discussed by Ablin & Peyré (2022, §3.5).

4. Related Works

Riemannian optimization. Many works try to improve the scalability of RGD (Absil et al., 2008) without giving up on feasibility (**D1**). As such, these methods depend one way or another on the use of retractions whose scalability is limited. Among these we can find gradient-based (Abrudan et al., 2008), conjugate gradient (Abrudan et al., 2009) or trust-region methods (Absil et al., 2007). Interestingly, recent works improve scalability by optimizing a random submanifold at each step (Han et al., 2024; 2025).

Trivializations. An alternative line of research uses retractions as surjective functions to map unconstrained parameters as feasible solutions (Casado & Martínez-Rubio, 2019; Casado, 2019). Thus, these methods can leverage techniques in Euclidean space, e.g. Adam (Kingma & Ba, 2015). Unfortunately, numerical and scalability issues inherent from retractions still persist with these approaches.

Infeasible methods trade scalability with feasibility, allowing solutions slightly violate constraints and converge

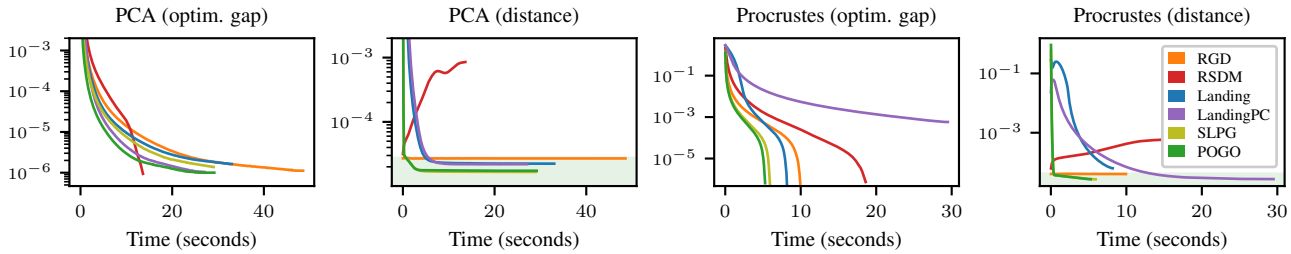

Figure 4. **POGO reduces the optimality gap the fastest across all baselines while staying on the manifold**. Results are averaged over 10 independent runs and the orthogonal matrices are of size 1500×2000 for PCA and 2000×2000 for Procrustes.

back to feasible solutions. Among these methods we find Lagrangian-based methods (Gao et al., 2019; 2022), as well as Landing and its variants (Ablin & Peyré, 2022; Ablin et al., 2024; Vary et al., 2024; Song et al., 2025; Loconte et al., 2026). Finally, we can also find proximal based methods for non-smooth optimization (Chen et al., 2020; 2021). In particular, Liu et al. (2024) propose a sequential linearized proximal gradient method (SLPG) where at each iteration it applies an approximate orthogonalization step. When they approximate it with a Taylor expansion, their normal step coincides with ours when $\lambda = 1/2$, and one could recover the POGO update in the case where $p \in \{1, n\}$. For an in-depth comparison, refer to App. C.

5. POGO in Action

We now evaluate the performance of POGO in relation with existing baselines. Namely, we are interested in verifying the three desiderata for an orthoptimizer described in §1, i.e.: **D1)** stay *close to the manifold*; **D2)** be computationally efficient; and **D3)** obtain a good *downstream performance*. While here we present the main results, experimental details and additional results can be found in App. D.

Baselines. We compare POGO to the following methods: **i)** Riemannian Gradient Descent (RGD) with QR retraction (Absil et al., 2008), Riemannian Random Submanifold Descent (RSDM) with orthogonal sampling (Han et al., 2025), Landing (Ablin & Peyré, 2022), LandingPC (Loconte et al., 2026), and SLPG (Liu et al., 2024). Unless otherwise stated, we consider POGO with λ set to $1/2$ and $\boldsymbol{G} = \nabla f(\boldsymbol{X})$, i.e., directly plugging the Euclidean gradient. Note that some baselines serve as quasi-ablations, e.g., RGD ablates the polar-retraction approximation (§3.3) and SLPG the use of adaptive base optimizers (§3.1 and App. C).

5.1. Single-Matrix Optimization

We focus first on ML problems involving one single orthogonal matrix, conforming a simple testbed where we can study the orthoptimizers in ideal conditions.

To this end, we follow the setup of (Han et al., 2025) and reproduce the largest experiments therein. Additionally, we do 10 repetitions per experiment and report the average.

Online PCA. We consider first the problem of finding the largest eigenvectors of a given matrix $\boldsymbol{A} \in \mathbb{R}^{n \times n}$,

$$\max_{\boldsymbol{X} \in \mathbb{R}^{p \times n}} \|\boldsymbol{X}\boldsymbol{A}\|^2 \quad \text{such that } \boldsymbol{X} \in \mathrm{St}(p, n)\,. \tag{15}$$

We set $n = 2000$, $p = 1500$ and, like Han et al. (2025), initialize $\boldsymbol{A}\boldsymbol{A}^\top$ as a positive definite matrix with a condition number of 1000 and exponentially decaying eigenvalues. For RSDM we set the submanifold dimension to 700. As the analytical solution of Eq. 15 is the top-p eigenvectors of $\boldsymbol{A}$, we use as performance metric the optimality gap, i.e., the relative error between the optimal loss and that found by the orthoptimizer. We train all methods by 3000 iterations and early stop them if they reach an optimality gap of 1×10^{-6}, reporting the training time until then and the distance of $\boldsymbol{X}$ to the Stiefel manifold as measured by $\|\boldsymbol{X}\boldsymbol{X}^\top - \boldsymbol{I}\|$.

We can observe the evolution of the optimality gap and manifold distance on the two left-most plots of Fig. 4. There, we see that POGO and LandingPC converge first, with Landing, SLPG and RGD descending at a similar rate. Remarkably, RSDM has the slowest start and then sharply descends. Regarding feasibility, we see every method quickly landing onto the manifold (green area, SLPG overlaps with POGO), except for RSDM which instead seems to consistently move away from the manifold. App. D.5 shows that the latter is solved using (slow) 64-bit floating-point arithmetic.

Orthogonal Procrustes problem. Next, we focus on the problem of matrix alignment (Gower & Dijksterhuis, 2004): Given two matrices $\boldsymbol{A} \in \mathbb{R}^{p \times p}$ and $\boldsymbol{B} \in \mathbb{R}^{p \times n}$, solve

$$\min_{\boldsymbol{X} \in \mathbb{R}^{p \times n}} \|\boldsymbol{A}\boldsymbol{X} - \boldsymbol{B}\|^2 \quad \text{such that } \boldsymbol{X} \in \mathrm{St}(p, n)\,. \tag{16}$$

We set $p = n = 2000$, initialize all the entries of $\boldsymbol{A}$ and $\boldsymbol{B}$ with independent standard Gaussian samples, and set the submanifold dimension of RSDM to 900. Just like before, we know that the analytical solution is the projection onto the Stiefel manifold of the matrix $\boldsymbol{A}\boldsymbol{B}$, and thus evaluate performance as the optimality gap. This time, we train each method for a maximum of 3000 iterations with early

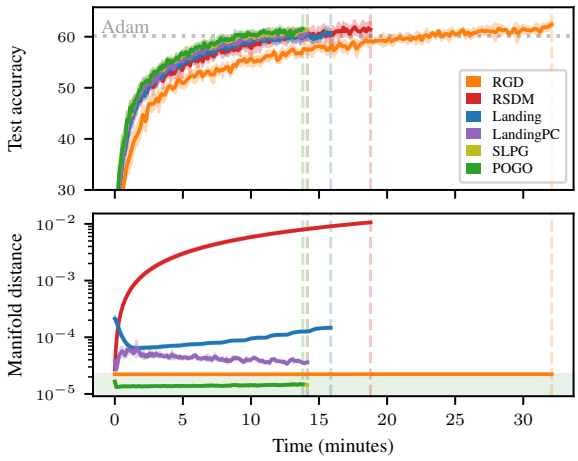

Figure 5. While all methods obtain similar test accuracies with an O-ViT (Fei et al., 2022) on CIFAR-10 (Krizhevsky et al., 2009), **POGO is the fastest method to complete** 10 **epochs without leaving the manifold**. Results show average and 95 % confidence intervals over 5 independent runs.

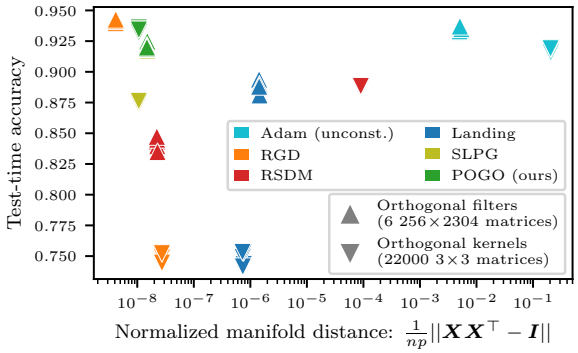

Figure 6. Normalized distance and test-time accuracy on the CNN experiment from §5.2. We observe that **in all instances POGO obtains similar accuracy to the unconstrained baseline while staying on the Stiefel manifold**.

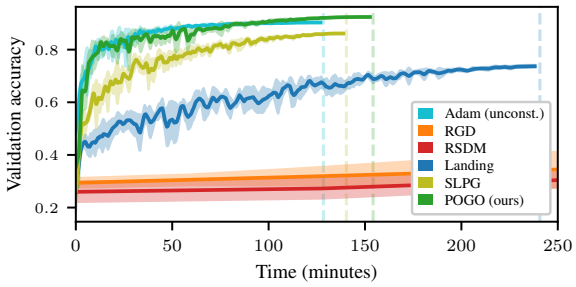

Figure 7. Evolution of the accuracy for the CNN experiment with orthogonal kernels. **POGO can learn at the same pace as the unconstrained Adam baseline despite orthogonality constraints.**

stopping if the optimality gap reaches a value of 1×10^{-6}.

The two rightmost plots of Fig. 4 show the optimality gap and manifold distance of all methods during training, where we can observe that POGO and SLPG converge significantly quicker, and that LandingPC exhausts all the training iterations. It is worth noting that the learning rate of LandingPC was set to the maximum allowed to safely bound the manifold distance. Finally, we observe that POGO and SLPG immediately go to the manifold, whereas both landing methods take considerably longer. Once again, we observe that RSDM consistently strays away from the manifold.

5.2. Multiple-Matrix Optimization

We just saw that POGO meets our desiderata (**D1-3**) on experiments with one matrix. Now we move to cases where orthogonality is imposed to neural-network parameters. This is ultimately our setting of interest and where we expect the most challenges, as they involve potentially many matrices that interact between them.

Vision Transformers. First, we consider a classification problem on CIFAR-10 (Krizhevsky et al., 2009) using a 28 M-parameter Orthogonal Vision Transformer (O-ViT) (Fei et al., 2022), a ViT with orthogonal attention matrices. To this end, we follow the setting of Han et al. (2025) and train a small-size O-ViT for 10 epochs with all methods, independently repeating the experiment five times. In this setting, we have 18 matrices with $n = p = 1024$ and we set the submanifold of RSDM to 300 as in the original paper.

Fig. 5 shows the training curves of all methods w.r.t. test accuracy and manifold distance. All methods achieve similar test accuracy, slightly surpassing an unconstrained baseline trained with Adam for reference (gray dotted line). Thus, the main difference becomes the training speed, where we observe that both landing methods and POGO are twice as fast as RGD, and POGO finishes 5 minutes quicker than RSDM.

We observe bigger discrepancies on the manifold distances. Namely, RSDM leaves the manifold again, this time *3 orders of magnitude* further than the reference method, RGD. Most notably, POGO stays in the manifold all training, as its distance is always lower than that of RGD, the retraction method. Interestingly, we see that Landing initially gets closer to the manifold and then leaves it again where its extension, LandingPC, mirrors this behavior: first increases its distance, and then consistently gets closer to the manifold.

Convolutional neural networks. To test the scalability of POGO (**D2**), we now consider a 2 M-parameter CNN tailored for CIFAR-10 classification (Jordan, 2024). Let $O \times I \times k \times k$ be the parameter dimensions of each convolutional layer, where O and I are the output and input dimensions, and k is the kernel size. Then, we devise two experiments. First, we follow prior works (Han et al., 2025; Ablin et al., 2024) and consider *orthogonal filters* of size $O \times Ik^2$ by flattening the tensor, resulting in 6 matrices with sizes ranging from 64×216 up to 256×2304, similar to the O-ViT case. Second, we assume instead *orthogonal kernels*

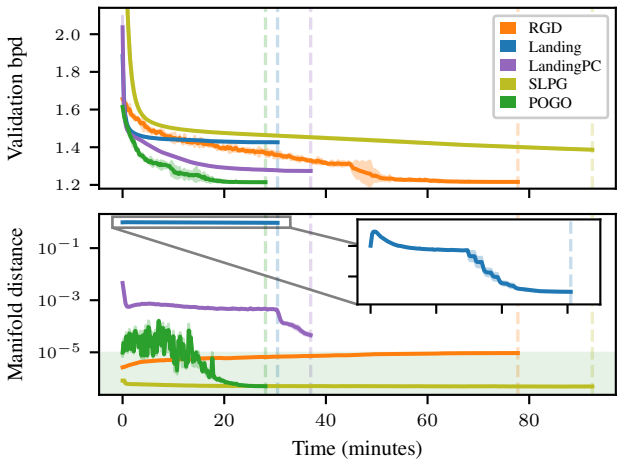

Figure 8. Training curves for validation bits-per-dimension and manifold distance for the squared unitary PC experiment. **POGO obtains state-of-the-art results for this class of models, while converging the fastest and not leaving the Stiefel manifold.**

(Ozay & Okatani, 2016) of size $k \times k$, leading to OI orthogonal matrices per layer and, in total, 218624 matrices of size 3×3. In both cases, we train for 100 epochs, set the submanifold dimension of RSDM to 64 for orthogonal filters, and 2 for orthogonal kernels, and use POGO with VAdam.

We present the results in Figs. 1 and 6, where training time is the most remarkable metric: while POGO takes 3 minutes to train in both experiments, RGD and RSDM take double that time with orthogonal filters, and *several hours* more with orthogonal kernels. This not only highlights the scalability of POGO, but also the lack of it for methods that rely on the QR algorithm (Francis, 1961; Kublanovskaya, 1962). We attribute the time differences with Landing to implementation details and extra computations for the learning rate.

Most remarkably, we observe that POGO consistently obtains accuracy comparable to Adam with minimal extra overhead (**D3**). Also, while SLPG matches POGO in the orthogonal-filter case (both overlap), it does not match its performance in the orthogonal-kernel case, as we had to train SLPG with very low learning rates to avoid numerical errors. This can be also appreciated in the training curves shown in Fig. 7. Finally, as shown in Fig. 6, these results are with no compromise in feasibility (**D1**): As we make the manifold dimension-invariant, we observe that most methods reach consistent distances (except for RSDM), and that POGO and SLPG stay in every case within the manifold.

5.3. Large-Scale Orthogonality Constraints for Tractable Inference

In this last section, we introduce a new DL benchmark where *large-scale orthogonality constraints are required*, thus serving as further motivation for orthoptimizers. An additional ablation study comparing POGO with different learning rates and both policies for λ can be found in App. D.6.

Squared unitary PCs. Recently, Loconte et al. (2026) introduced a new class of probabilistic circuits (PCs) (Choi et al., 2020; Vergari et al., 2021) called *squared unitary PCs*. In summary, squared unitary PCs are expressive models which, by imposing orthogonality constraints to their parameters, ensure that one can obtain an already-normalized probability distribution after multiplying the PC with itself. For our purposes, squared unitary PCs constitute a great testbed for orthoptimizers not only due to their scalability (up to 300 M parameters, **D2**), but also because they represent an example where orthogonality is needed in practice: Due to prohibitively large memory requirements, using unconstrained parameters and re-normalizing the squared PC is infeasible in practice.

In fact, to train squared unitary PCs Loconte et al. (2026) introduced LandingPC as a variant of Landing tailored for these models, and thus it represents the current SotA. Moreover, these models can work with complex parameters, serving us an opportunity to test POGO in the complex Stiefel manifold. Therefore, we follow an identical setup as Loconte et al. (2026) and simply use each optimizer as drop-in replacements of LandingPC.

Experimental setting. We train squared unitary PCs with 10 input units to fit the empirical distribution of the MNIST dataset (LeCun et al., 2010), resulting in 1048 complex-valued matrices ranging from 10×256 up to 10×10000. Following (Loconte et al., 2026), every model is trained a maximum of 200 epochs, and use early stopping based on the validation set as well as halve the learning rate when the loss plateaus for 10 epochs. Additionally, we remove RSDM from the plots as we did not manage to obtain anything close to the performance of other orthoptimizers.

Results. Fig. 8 shows the downstream performance in bits-per-dimension (lower is better) and manifold distance for all methods during training. Here, we can observe that RGD obtains quite good performance, although it takes it one hour and a half to do so. We can also observe a clear difference between Landing and LandingPC: The latter obtains great performance and consistently nears the manifold, ending nearby *before early stopping kicks in*. In contrast, Landing gets stuck early on at $\varepsilon = 0.5$ and the inset plot shows that it spends the rest of training slowly reducing the distance to the manifold. Similar to the previous case, we had to set an extremely low learning rate for SLPG not to crash, which explains its slow convergence and great feasibility. Finally, POGO converges extremely quickly while staying really close to the manifold. *Compared with RGD, POGO obtains the same performance while being twice as fast.*

6. Concluding Remarks

In this work, we substantially advanced the current SoTA of orthoptimizers while preserving three key properties (**D1-3**). We have done so by introducing our POGO, which refines the ideas of Landing (Ablin & Peyré, 2022), resulting in an orthoptimizer which accurately approximates the optimal step size to land on the Stiefel manifold after each gradient update, and thus respecting orthogonality constraints closely. POGO raises the bar of how orthoptimizers can scale to real-world deep learning problems with thousands of matrices and of what they achieve in terms of dowstream performance, being finally competitive with unconstrained optimizers. We foresee that POGO can open up several applications requiring orthogonality constraints, from quantum computing (Orús, 2019) to physics applications (Cai & White, 2008), that have been unexplored so far.

Acknowledgments

We acknowledge Lorenzo Loconte for his helpful feedback at the first steps of the project, Pierre Ablin for his feedback early in development, and to the april lab for their support. Both authors are supported by the "UNREAL: Unified Reasoning Layer for Trustworthy ML" project (EP/Y023838/1) selected by the ERC and funded by UKRI EPSRC.

Authors Contributions

AJ conceived the initial idea and discussed it with AV. AJ is responsible for all the theoretical contributions, plots and experiments. AJ led the writing with help from AV who provided feedback at every stage of the project .

Reproducibility Statement

All hyperparameter values and experimental settings are detailed in the appendices and properly referenced in the main text. The code to reproduce the experiments in this paper can be found in github.com/april-tools/pogo-experiments.

Impact Statement

This paper presents work whose goal is to advance the field of machine learning optimization. There are many potential societal consequences of our work, none of which we feel must be specifically highlighted here.

References

Abdelfattah, A. and Fasi, M. An Efficient Batch Solver for the Singular Value Decomposition on GPUs. *ArXiv preprint*, abs/2601.17979, 2026. URL https://arxiv.org/abs/2601.17979. (Cited in page 5.)

Ablin, P. and Peyré, G. Fast and accurate optimization on the orthogonal manifold without retraction. In Camps-Valls, G., Ruiz, F. J. R., and Valera, I. (eds.), *International Conference on Artificial Intelligence and Statistics, AISTATS 2022, 28-30 March 2022, Virtual Event*, volume 151 of *Proceedings of Machine Learning Research*, pp. 5636–5657. PMLR, 2022. URL https://proceedings.mlr.press/v151/ablin22a.html. (Cited in pages 1, 2, 3, 5, 6, and 9.)

Ablin, P., Vary, S., Gao, B., and Absil, P.-A. Infeasible deterministic, stochastic, and variance-reduction algorithms for optimization under orthogonality constraints. *Journal of Machine Learning Research*, 25(389):1–38, 2024. (Cited in pages 1, 2, 3, 6, and 7.)

Abrudan, T., Eriksson, J., and Koivunen, V. Conjugate gradient algorithm for optimization under unitary matrix constraint. *Signal Processing*, 89(9):1704–1714, 2009. (Cited in page 5.)

Abrudan, T. E., Eriksson, J., and Koivunen, V. Steepest descent algorithms for optimization under unitary matrix constraint. *IEEE Transactions on Signal Processing*, 56 (3):1134–1147, 2008. (Cited in page 5.)

Absil, P.-A., Baker, C. G., and Gallivan, K. A. Trust-region methods on Riemannian manifolds. *Foundations of Computational Mathematics*, 7(3):303–330, 2007. (Cited in page 5.)

Absil, P.-A., Mahony, R., and Sepulchre, R. *Optimization algorithms on matrix manifolds*. Princeton University Press, 2008. (Cited in pages 1, 2, 5, and 6.)

Arioli, M., Codenotti, B., and Fassino, C. The Padé method for computing the matrix exponential. *Linear algebra and its applications*, 240:111–130, 1996. (Cited in page 2.)

Arjovsky, M., Shah, A., and Bengio, Y. Unitary Evolution Recurrent Neural Networks. In Balcan, M. and Weinberger, K. Q. (eds.), *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, volume 48 of *JMLR Workshop and Conference Proceedings*, pp. 1120–1128. JMLR.org, 2016. URL http://proceedings.mlr.press/v48/arjovsky16.html. (Cited in pages 1 and 2.)

Bansal, N., Chen, X., and Wang, Z. Can We Gain More from Orthogonality Regularizations in Training Deep Networks? In Bengio, S., Wallach, H. M., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pp. 4266–4276, 2018. URL https://proceedings.neurips.cc/paper/2018/hash/bf424cb7b0dea050a42b9739eb261a3a-Abstract.html. (Cited in page 2.)

Bishop, C. *Pattern Recognition and Machine Learning*. Springer, 2006. URL https://www.microsoft.com/en-us/research/publication/pattern-recognition-machine-learning/. (Cited in page 1.)

Boumal, N. *An introduction to optimization on smooth manifolds*. Cambridge University Press, 2023. (Cited in page 5.)

Cai, L. and White, R. E. Reduction of model order based on proper orthogonal decomposition for lithium-ion battery simulations. *Journal of the Electrochemical Society*, 156 (3):A154, 2008. (Cited in page 9.)

Cardoso, J.-F. and Laheld, B. H. Equivariant adaptive source separation. *IEEE Transactions on signal processing*, 44 (12):3017–3030, 2002. (Cited in page 2.)

Casado, M. L. Trivializations for Gradient-Based Optimization on Manifolds. In Wallach, H. M., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E. B., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pp. 9154–9164, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/1b33d16fc562464579b7199ca3114982-Abstract.html. (Cited in page 5.)

Casado, M. L. and Martínez-Rubio, D. Cheap Orthogonal Constraints in Neural Networks: A Simple Parametrization of the Orthogonal and Unitary Group. In Chaudhuri, K. and Salakhutdinov, R. (eds.), *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pp. 3794–3803. PMLR, 2019. URL http://proceedings.mlr.press/v97/lezcano-casado19a.html. (Cited in page 5.)

Chen, S., Ma, S., Man-Cho So, A., and Zhang, T. Proximal gradient method for nonsmooth optimization over the Stiefel manifold. *SIAM Journal on Optimization*, 30(1): 210–239, 2020. (Cited in page 6.)

Chen, S., Deng, Z., Ma, S., and So, A. M.-C. Manifold proximal point algorithms for dual principal component pursuit and orthogonal dictionary learning. *IEEE transactions on signal processing*, 69:4759–4773, 2021. (Cited in page 6.)

Choi, Y., Vergari, A., and Van den Broeck, G. Probabilistic Circuits: A Unifying Framework for Tractable Probabilistic Modeling. Technical report, University of California, Los Angeles (UCLA), 2020. (Cited in pages 8 and 22.)

Dongarra, J., Gates, M., Haidar, A., Kurzak, J., Luszczek, P., Tomov, S., and Yamazaki, I. The Singular Value Decomposition: Anatomy of Optimizing an Algorithm for Extreme Scale. *SIAM Review*, 60(4):808–865, 2018. doi: 10.1137/17M1117732. URL https://doi.org/10.1137/17M1117732. (Cited in page 3.)

Edelman, A., Arias, T. A., and Smith, S. T. The geometry of algorithms with orthogonality constraints. *SIAM journal on Matrix Analysis and Applications*, 20(2):303–353, 1998. (Cited in page 2.)

Fei, Y., Liu, Y., Wei, X., and Chen, M. O-vit: Orthogonal vision transformer. *ArXiv preprint*, abs/2201.12133, 2022. URL https://arxiv.org/abs/2201.12133. (Cited in pages 1, 2, and 7.)

Francis, J. G. F. The QR Transformation A Unitary Analogue to the LR Transformation–Part 1. *The Computer Journal*, 4(3):265–271, 1961. ISSN 0010-4620. doi: 10.1093/comjnl/4.3.265. URL https://doi.org/10.1093/comjnl/4.3.265. (Cited in page 8.)

Gao, B., Liu, X., and Yuan, Y.-x. Parallelizable algorithms for optimization problems with orthogonality constraints. *SIAM Journal on Scientific Computing*, 41(3):A1949–A1983, 2019. (Cited in page 6.)

Gao, B., Hu, G., Kuang, Y., and Liu, X. An orthogonalization-free parallelizable framework for all-electron calculations in density functional theory. *SIAM Journal on Scientific Computing*, 44(3):B723–B745, 2022. (Cited in page 6.)

Goliński, A., Lezcano-Casado, M., and Rainforth, T. Improving Normalizing Flows via Better Orthogonal Parameterizations. In *ICML Workshop on Invertible Neural Networks*, 2019. URL https://api.semanticscholar.org/CorpusID:271877803. (Cited in page 2.)

Gower, J. C. and Dijksterhuis, G. B. *Procrustes problems*, volume 30. Oxford university press, 2004. (Cited in page 6.)

Han, A., Jawanpuria, P., and Mishra, B. Riemannian coordinate descent algorithms on matrix manifolds. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=bdKaQmrM81. (Cited in page 5.)

Han, A., Poirion, P.-L., and Takeda, A. Efficient Optimization with Orthogonality Constraint: a Randomized Riemannian Submanifold Method. *ArXiv preprint*, abs/2505.12378, 2025. URL https://arxiv.org/abs/2505.12378. (Cited in pages 1, 2, 5, 6, and 7.)

Henry, G., Tang, P. T. P., and Heinecke, A. Leveraging the bfloat16 artificial intelligence datatype for higher-precision computations. In *2019 IEEE 26th Symposium on Computer Arithmetic (ARITH)*, pp. 69–76. IEEE, 2019. (Cited in page 5.)

Hyvärinen, A., Hurri, J., and Hoyer, P. O. Independent component analysis. In *Natural Image Statistics: A Probabilistic Approach to Early Computational Vision*, pp. 151–175. Springer, 2001. (Cited in page 1.)

Javaloy, A. and Valera, I. RotoGrad: Gradient Homogenization in Multitask Learning. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022. URL https://openreview.net/forum?id=T8wHz4rnuGL. (Cited in page 1.)

Jordan, K. 94% on CIFAR-10 in 3.29 Seconds on a Single GPU. *ArXiv preprint*, abs/2404.00498, 2024. URL https://arxiv.org/abs/2404.00498. (Cited in pages 1 and 7.)

Jordan, K., Jin, Y., Boza, V., You, J., Cesista, F., Newhouse, L., and Bernstein, J. Muon: An optimizer for hidden layers in neural networks, 2024. URL https://kellerjordan.github.io/posts/muon/. (Cited in page 2.)

Kiani, B. T., Balestriero, R., LeCun, Y., and Lloyd, S. projUNN: efficient method for training deep networks with unitary matrices. In Koyejo, S., Mohamed, S., Agarwal, A., Belgrave, D., Cho, K., and Oh, A. (eds.), *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022. URL http://papers.nips.cc/paper_files/paper/2022/hash/5d1a0188e18c1d74a0f8d6eb5ecede4f-Abstract-Conference.html. (Cited in page 1.)

Kingma, D. P. and Ba, J. Adam: A Method for Stochastic Optimization. In Bengio, Y. and LeCun, Y. (eds.), *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL http://arxiv.org/abs/1412.6980. (Cited in pages 2, 4, and 5.)

Krizhevsky, A., Hinton, G., et al. Learning multiple layers of features from tiny images. 2009. (Cited in pages 1 and 7.)

Kublanovskaya, V. N. On some algorithms for the solution of the complete eigenvalue problem. *USSR Computational Mathematics and Mathematical Physics*, 1(3):637–657, 1962. (Cited in page 8.)

LeCun, Y., Cortes, C., and Burges, C. MNIST handwritten digit database. *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*, 2, 2010. (Cited in page 8.)

Li, S., Jia, K., Wen, Y., Liu, T., and Tao, D. Orthogonal deep neural networks. *IEEE transactions on pattern analysis and machine intelligence*, 43(4):1352–1368, 2019. (Cited in page 1.)

Ling, S., Sharp, N., and Jacobson, A. VectorAdam for Rotation Equivariant Geometry Optimization. In Koyejo, S., Mohamed, S., Agarwal, A., Belgrave, D., Cho, K., and Oh, A. (eds.), *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022. URL http://papers.nips.cc/paper_files/paper/2022/hash/1a774f3555593986d7d95e4780d9e4f4-Abstract-Conference.html. (Cited in pages 4, 5, 19, and 24.)

Liu, X., Xiao, N., and Yuan, Y.-x. A penalty-free infeasible approach for a class of nonsmooth optimization problems over the Stiefel manifold. *Journal of Scientific Computing*, 99(2):30, 2024. (Cited in pages 5, 6, 16, and 20.)

Loconte, L., Sladek, A. M., Mengel, S., Trapp, M., Solin, A., Gillis, N., and Vergari, A. Subtractive Mixture Models via Squaring: Representation and Learning. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=xIHi5nxu9P. (Cited in page 22.)

Loconte, L., Mengel, S., and Vergari, A. Sum of squares circuits. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 19077–19085, 2025. (Cited in page 22.)

Loconte, L., Javaloy, A., and Vergari, A. How to Square Tensor Networks and Circuits Without Squaring Them. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=gHPRSPxIsk. (Cited in pages 1, 2, 6, 8, and 22.)

Moler, C. and Loan, C. Nineteen Dubious Ways to Compute the Exponential of a Matrix, Twenty-Five Years Later. *SIAM Review*, 45:3–49, 2006. doi: 10.1137/S0036144502418024180. (Cited in page 2.)

Morgan, N. and Bourlard, H. Generalization and parameter estimation in feedforward nets: Some experiments. *Advances in neural information processing systems*, 2, 1989. (Cited in page 3.)

Orús, R. Tensor networks for complex quantum systems. *Nature Reviews Physics*, 1(9):538–550, 2019. (Cited in page 9.)

Ozay, M. and Okatani, T. Optimization on submanifolds of convolution kernels in cnns. *ArXiv preprint*, abs/1610.07008, 2016. URL https://arxiv.org/abs/1610.07008. (Cited in pages 2 and 8.)

Song, Y., Li, P., Gao, B., and Yuan, K. Distributed Retraction-Free and Communication-Efficient Optimization on the Stiefel Manifold. *ArXiv preprint*, abs/2506.02879, 2025. URL https://arxiv.org/abs/2506.02879. (Cited in pages 2 and 6.)

Stiefel, E. *Richtungsfelder und Fernparallelismus in n-dimensionalen Mannigfaltigkeiten*. PhD thesis, ETH Zurich, 1935. (Cited in page 2.)

Vary, S., Ablin, P., Gao, B., and Absil, P. Optimization without Retraction on the Random Generalized Stiefel Manifold. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=QLtxj3erlJ. (Cited in pages 1 and 6.)

Vergari, A., Choi, Y., Liu, A., Teso, S., and den Broeck, G. V. A Compositional Atlas of Tractable Circuit Operations for Probabilistic Inference. In Ranzato, M., Beygelzimer, A., Dauphin, Y. N., Liang, P., and Vaughan, J. W. (eds.), *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pp. 13189–13201, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/6e01383fd96a17ae51cc3e15447e7533-Abstract.html. (Cited in pages 8 and 22.)

Wang, J., Chen, Y., Chakraborty, R., and Yu, S. X. Orthogonal Convolutional Neural Networks. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pp. 11502–11512. IEEE, 2020. doi: 10.1109/CVPR42600.2020.01152. URL https://doi.org/10.1109/CVPR42600.2020.01152. (Cited in pages 1 and 2.)

Appendix

Table of Contents

A. Theoretical Results

A.1. Auxiliary Results

Lemma A.1. *If* $\boldsymbol{X} \in \mathrm{St}^{\varepsilon}(p,n)$ *then* $\sqrt{\max(0, 1-\varepsilon/\sqrt{p})} \leq \|\boldsymbol{X}\|_2 \leq \sqrt{1+\varepsilon/\sqrt{p}}$.

Proof. Assume $\boldsymbol{X} \in \mathrm{St}^{\varepsilon}(p,n)$ and let us denote by $\sigma_i(\boldsymbol{X})$ the i-th singular value of $\boldsymbol{X}$. Then, $\|\boldsymbol{X}\boldsymbol{X}^\top - \boldsymbol{I}\| \leq \varepsilon$.

$$\begin{aligned}
\|\boldsymbol{X}\boldsymbol{X}^\top - \boldsymbol{I}\|^2 &= \|\boldsymbol{X}\boldsymbol{X}^\top\|^4 + \|\boldsymbol{I}\|^2 - 2\langle \boldsymbol{X}\boldsymbol{X}^\top, \boldsymbol{I}\rangle && (17)\\
&= \sum_i \sigma_i(\boldsymbol{X})^2 + \sum_i 1 - 2\,\mathrm{Tr}(\boldsymbol{I}^\top \boldsymbol{X}\boldsymbol{X}^\top) && (18)\\
&= \sum_i \sigma_i(\boldsymbol{X})^2 + \sum_i 1 - 2\sum_i \sigma_i(\boldsymbol{X})^2 && (19)\\
&= \sum_i \left(\sigma_i(\boldsymbol{X})^4 + 1 - 2\sigma_i(\boldsymbol{X})^2\right) && (20)\\
&= \sum_i \left(\sigma_i(\boldsymbol{X})^2 - 1\right)^2 \leq \varepsilon^2 && (21)
\end{aligned}$$

Therefore

$$\begin{aligned}
&\sum_i \left(\sigma_i(\boldsymbol{X})^2 - 1\right)^2 \leq p\left(\|\boldsymbol{X}\|_2^2 - 1\right)^2 \leq \varepsilon^2 \Rightarrow \sqrt{p}|\|\boldsymbol{X}\|_2^2 - 1| \leq \varepsilon && (22)\\
&-\varepsilon/\sqrt{p} \leq \|\boldsymbol{X}\|_2^2 - 1 \leq \varepsilon/\sqrt{p} \Rightarrow 1 - \varepsilon/\sqrt{p} \leq \|\boldsymbol{X}\|_2^2 \leq 1 + \varepsilon/\sqrt{p} && (23)\\
&\sqrt{\max(0, 1-\varepsilon/\sqrt{p})} \leq \|\boldsymbol{X}\|_2 \leq \sqrt{1+\varepsilon/\sqrt{p}} && (24)
\end{aligned}$$

□

Proposition A.2. *If* $\boldsymbol{X} \in \mathrm{St}^{\varepsilon}(p,n)$ *then we can write* $\boldsymbol{X} = \boldsymbol{U}(\boldsymbol{I} + \boldsymbol{\Delta})\boldsymbol{V}^\top$ *where* $\|\boldsymbol{\Delta}\| \leq \varepsilon$ *and* $\boldsymbol{U}, \boldsymbol{V} \in \mathrm{St}(p,n)$.

Proof. Using Lemma A.1 we know that each singular value of $\boldsymbol{X}$ can be written as $\sigma_i(\boldsymbol{X}) = \sqrt{1+\upsilon_i}$ where $|\upsilon_i| \leq \hat{\varepsilon} := \varepsilon\sqrt{p}$.
We can rewrite $\sigma_i(\boldsymbol{X})$ as $\sigma_i(\boldsymbol{X}) = \pm 1 + \sqrt{1+\upsilon_i} = 1 + (\sqrt{1+\upsilon_i} - 1) = 1 + \delta_i$ and bound δ_i:

$$\sqrt{\max(0, 1-\hat{\varepsilon})} - 1 \leq \delta_i \leq \sqrt{1+\hat{\varepsilon}} - 1 \tag{25}$$

And, since $\sqrt{\max(0, 1+x)} - 1 \leq x$ if $x \geq 0$ and $x \leq \sqrt{\max(0, 1+x)} - 1$ otherwise:

$$-\hat{\varepsilon} \leq \sqrt{\max(0, 1-\hat{\varepsilon})} - 1 \leq \delta_i \leq \sqrt{1+\hat{\varepsilon}} - 1 \leq \hat{\varepsilon}\,. \tag{26}$$

We can thus rewrite the singular values of $\boldsymbol{X}$ as $\sigma_i(\boldsymbol{X}) = 1 + \delta_i$ with $|\delta_i| \leq \hat{\varepsilon}$ and its singular value decomposition as $\boldsymbol{X} = \boldsymbol{U}(\boldsymbol{I} + \boldsymbol{\Delta})\boldsymbol{V}^\top$ with $\|\boldsymbol{\Delta}\| \leq \sqrt{p}\|\boldsymbol{\Delta}\|_2 = \hat{\varepsilon}\sqrt{p} = \varepsilon$. □

Proposition A.3. *For any square matrix* $\boldsymbol{X}$*, we always have that* $\|\mathrm{Skew}\,\boldsymbol{X}\| \leq \|\boldsymbol{X}\|$ *and* $\|\mathrm{Sym}\,\boldsymbol{X}\| \leq \|\boldsymbol{X}\|$.

Proof. The space of square matrices decomposes as the direct sum of symmetric and skew-symmetric matrices, i.e., $M_p = \mathrm{Sym}_p \oplus \mathrm{Skew}_p$. Call $\mathrm{Sym}(\boldsymbol{X}) = \frac{1}{2}(\boldsymbol{X} + \boldsymbol{X}^\top)$ and $\mathrm{Skew}(\boldsymbol{X}) = \frac{1}{2}(\boldsymbol{X} - \boldsymbol{X}^\top)$. Then, for any square matrix $\boldsymbol{X}$:

$$\boldsymbol{X} = \frac{2}{2}(\boldsymbol{X} \pm \boldsymbol{X}^\top) = \frac{1}{2}(\boldsymbol{X} + \boldsymbol{X}^\top) + \frac{1}{2}(\boldsymbol{X} - \boldsymbol{X}^\top) = \mathrm{Sym}(\boldsymbol{X}) + \mathrm{Skew}(\boldsymbol{X})\,, \tag{27}$$

and therefore

$$\begin{aligned}
\|\boldsymbol{X}\|^2 &= \|\mathrm{Sym}(\boldsymbol{X})\|^2 + \|\mathrm{Skew}(\boldsymbol{X})\|^2 + 2\langle \mathrm{Sym}(\boldsymbol{X}), \mathrm{Skew}(\boldsymbol{X})\rangle && (28)\\
\|\mathrm{Sym}(\boldsymbol{X})\|^2 &= \|\boldsymbol{X}\|^2 - \|\mathrm{Skew}(\boldsymbol{X})\|^2 \leq \|\boldsymbol{X}\|^2 \Rightarrow \|\mathrm{Sym}(\boldsymbol{X})\| \leq \|\boldsymbol{X}\| && (29)
\end{aligned}$$

$$\|\mathrm{Skew}(\boldsymbol{X})\|^2 = \|\boldsymbol{X}\|^2 - \|\mathrm{Sym}(\boldsymbol{X})\|^2 \leq \|\boldsymbol{X}\|^2 \Rightarrow \|\mathrm{Skew}(\boldsymbol{X})\| \leq \|\boldsymbol{X}\| \,. \tag{30}$$

□

Proposition A.4. *Assume that $\boldsymbol{X} \in \mathrm{St}(p,n)$, $\|\boldsymbol{G}\| \leq L$ and that $\xi \coloneqq \eta L < 1$, then $s_i = \eta^i \operatorname{Tr}(\boldsymbol{S}^i) \leq \xi^i$.*

Proof. We have the following inequality:

$$\eta^k \|\boldsymbol{S}^k\| \leq \eta^k \|\boldsymbol{S}\|^k \leq \eta^k \|\boldsymbol{X}^\top \boldsymbol{G}\|^k = \eta^k \|\boldsymbol{G}\|^k \leq \eta^k L^k = \xi^k \,, \tag{31}$$

where $\|\boldsymbol{S}\| \leq \|\boldsymbol{X}^\top \boldsymbol{G}\|$ comes from Prop. A.3 and $\|\boldsymbol{X}^\top \boldsymbol{G}\|^k = \|\boldsymbol{G}\|^k$ since $\boldsymbol{X} \in \mathrm{St}(p,n)$.
Then, we can assume $s_i = s_{2k}$ (since $s_i = 0$ for any odd i as $\boldsymbol{S}$ is skew-symmetric), and we have that

$$s_i = s_{2k} = \eta^{2k} \operatorname{Tr}(\boldsymbol{S}^{2k}) = \left(\eta^k \|\boldsymbol{S}^k\|\right)^2 \leq \xi^{2k} = \xi^i \,. \tag{32}$$

□

A.2. Proof of Lemma 3.1

Lemma A.5. *Let $\boldsymbol{X} \in \mathbb{R}^{p\times n}$, $\boldsymbol{M} = \boldsymbol{X} - \eta \boldsymbol{X}\boldsymbol{S}$ with $\eta > 0$ and $\boldsymbol{S} \in \mathbb{R}^{n\times n}$ skew-symmetric, and $\boldsymbol{X}_1 = \boldsymbol{M} - \lambda(\boldsymbol{M}\boldsymbol{M}^\top - \boldsymbol{I})\boldsymbol{M}$ with $\lambda > 0$. Then, the function $P(\lambda) = \mathcal{N}(\boldsymbol{X}_1) = \|\boldsymbol{X}_1\boldsymbol{X}_1^\top - \boldsymbol{I}\|^2$ is a quartic polynomial w.r.t. λ.*

Proof. Let us write $\boldsymbol{X}_1 = \boldsymbol{A} + \lambda \boldsymbol{B}$ with $\boldsymbol{A} = \boldsymbol{M}$ and $\boldsymbol{B} = -(\boldsymbol{M}\boldsymbol{M}^\top - \boldsymbol{I})\boldsymbol{M}$. Then, by expanding $P(\lambda)$ we get,

$$\begin{aligned} P(\lambda) &= \|(\boldsymbol{A} + \lambda\boldsymbol{B})(\boldsymbol{A} + \lambda\boldsymbol{B})^\top - \boldsymbol{I}_p\|^2 && (33)\\ &= \|\underbrace{(\boldsymbol{A}\boldsymbol{A}^\top - \boldsymbol{I}_p)}_{\boldsymbol{C}} + \underbrace{(\boldsymbol{A}\boldsymbol{B}^\top + \boldsymbol{B}\boldsymbol{A}^\top)}_{\boldsymbol{D}}\lambda + \underbrace{\boldsymbol{B}\boldsymbol{B}^\top}_{\boldsymbol{E}}\lambda^2\|^2 && (34)\end{aligned}$$

Now, let $\boldsymbol{C} \coloneqq \boldsymbol{A}_t\boldsymbol{A}_t^\top - \boldsymbol{I}$, $\boldsymbol{D} \coloneqq \boldsymbol{A}\boldsymbol{B}^\top + \boldsymbol{B}\boldsymbol{A}^\top$ and $\boldsymbol{E} \coloneqq \boldsymbol{B}\boldsymbol{B}^\top$. Then

$$\begin{aligned} P(\lambda) &= \|\boldsymbol{C} + \boldsymbol{D}\lambda + \boldsymbol{E}\lambda^2\|^2 && (35)\\ &= \operatorname{Tr}(\boldsymbol{E}^\top\boldsymbol{E})\lambda^4 + 2\operatorname{Tr}(\boldsymbol{D}^\top\boldsymbol{E})\lambda^3 + [\operatorname{Tr}(\boldsymbol{D}^\top\boldsymbol{D}) + 2\operatorname{Tr}(\boldsymbol{E}^\top\boldsymbol{D})]\lambda^2 + \operatorname{Tr}(\boldsymbol{C}^\top\boldsymbol{D})\lambda + \operatorname{Tr}(\boldsymbol{C}^\top\boldsymbol{C}) \,. && (36)\end{aligned}$$

□

A.3. Proof of Prop. 3.2

Proposition A.6. *Let $\boldsymbol{X} \in \mathrm{St}(p,n)$ and $\boldsymbol{M} = \boldsymbol{X} - \eta\boldsymbol{X}\boldsymbol{S}$ with $\boldsymbol{S}$ skew-symmetric, then*

$$\|\boldsymbol{M}\boldsymbol{M}^\top - \boldsymbol{I}\| \leq \eta^2 \|\boldsymbol{S}^2\| \,. \tag{37}$$

Proof. First, we simply $\boldsymbol{M}\boldsymbol{M}^\top - \boldsymbol{I}$,

$$\begin{aligned} \boldsymbol{M}\boldsymbol{M}^\top - \boldsymbol{I} &= ((\boldsymbol{X} - \eta\boldsymbol{X}\boldsymbol{S})(\boldsymbol{X} - \eta\boldsymbol{X}\boldsymbol{S})^\top - \boldsymbol{I}) && (38)\\ &= (\cancel{\boldsymbol{X}\boldsymbol{X}^\top - \boldsymbol{I}} - \eta\boldsymbol{X}\boldsymbol{S}^\top\boldsymbol{X}^\top - \eta\boldsymbol{X}\boldsymbol{S}\boldsymbol{X}^\top + \eta^2\boldsymbol{X}\boldsymbol{S}\boldsymbol{S}^\top\boldsymbol{X}^\top) && (39)\\ &= (\cancel{\eta\boldsymbol{X}\boldsymbol{S}\boldsymbol{X}^\top - \eta\boldsymbol{X}\boldsymbol{S}\boldsymbol{X}^\top} + \eta^2\boldsymbol{X}\boldsymbol{S}\boldsymbol{S}^\top\boldsymbol{X}^\top) && (40)\\ &= -\eta^2\boldsymbol{X}\boldsymbol{S}^2\boldsymbol{X}^\top \,, && (41)\end{aligned}$$

so that its squared norm is

$$\begin{aligned} \|\boldsymbol{M}\boldsymbol{M}^\top - \boldsymbol{I}\|^2 &= \eta^4\|-\boldsymbol{X}\boldsymbol{S}^2\boldsymbol{X}^\top\|^2 && (42)\\ &= \eta^4 \operatorname{Tr}(\boldsymbol{X}\boldsymbol{S}^2\boldsymbol{X}^\top\boldsymbol{X}\boldsymbol{S}^2\boldsymbol{X}^\top) && (43)\\ &= \eta^4 \operatorname{Tr}(\boldsymbol{S}^2\boldsymbol{P}\boldsymbol{S}^2\boldsymbol{P}) && (44)\\ &= \eta^4\|\boldsymbol{S}^2\boldsymbol{P}\|^2 && (45)\end{aligned}$$

$$\leq \eta^4 \|\boldsymbol{S}^2\|^2 \,, \tag{46}$$

where $\boldsymbol{P} \coloneqq \boldsymbol{X}^\top \boldsymbol{X}$ is the orthogonal projector of $\boldsymbol{X}$ and thus a bounded operator. Therefore,

$$\|\boldsymbol{M}\boldsymbol{M}^\top - \boldsymbol{I}\| \leq \eta^2 \|\boldsymbol{S}^2\| \,. \tag{47}$$

□

A.4. Proof of Prop. 3.3

Proposition A.7. *Let $\boldsymbol{X} \in \mathrm{St}(p,n)$, and let $P(\lambda) = \mathcal{N}(\boldsymbol{M} + \lambda(\boldsymbol{I} - \boldsymbol{M}\boldsymbol{M}^\top)\boldsymbol{M})$ be the landing polynomial of $\boldsymbol{X}$, with $\boldsymbol{M} = \boldsymbol{X} - \eta \boldsymbol{X}\boldsymbol{S}$, $\eta > 0$ and $\boldsymbol{S} = \mathrm{Skew}(\boldsymbol{X}^\top \boldsymbol{G})$. Assume also that $\|\boldsymbol{G}\| \leq L$ and let $\xi \coloneqq \eta L$. Then*

$$P(1/2) \leq \left(\frac{3}{4} + \frac{1}{4}\xi^2\right)^2 \xi^8 = \mathrm{o}(\xi^7) \quad \textit{as } \xi \to 0 \,. \tag{48}$$

Proof. Using Prop. A.6 we have that

$$\|\boldsymbol{M}\boldsymbol{M}^\top - \boldsymbol{I}\|^2 \leq \eta^4 \|\boldsymbol{S}^2\|^2 \leq \eta^4 \|\boldsymbol{G}\|^4 \leq (\eta L)^4 = \xi^4 \,. \tag{49}$$

Then, we can follow similar steps to those of Liu et al. (2024, Lemma 1):

$$\|\boldsymbol{I} - \frac{1}{4}\boldsymbol{M}\boldsymbol{M}^\top\| = \|\frac{3}{4}\boldsymbol{I} + \frac{1}{4}(\boldsymbol{I} - \boldsymbol{M}\boldsymbol{M}^\top)\| \leq \frac{3}{4} + \frac{1}{4}\|\boldsymbol{I} - \boldsymbol{M}\boldsymbol{M}^\top\| \leq \frac{3}{4} + \frac{1}{4}\xi^2 \,, \tag{50}$$

which helps us bound $P(\lambda)$ as

$$\begin{aligned}
\|\boldsymbol{X}_1\boldsymbol{X}_1^\top - \boldsymbol{I}\|^2 &= \|(\boldsymbol{M} + \lambda(\boldsymbol{I} - \boldsymbol{M}\boldsymbol{M}^\top)\boldsymbol{M})(\boldsymbol{M} + \lambda(\boldsymbol{I} - \boldsymbol{M}\boldsymbol{M}^\top)\boldsymbol{M})^\top - \boldsymbol{I}\|^2 && (51)\\
&= \|(\boldsymbol{M} + \lambda(\boldsymbol{I} - \boldsymbol{M}\boldsymbol{M}^\top)\boldsymbol{M})(\boldsymbol{M}^\top + \lambda\boldsymbol{M}^\top(\boldsymbol{I} - \boldsymbol{M}\boldsymbol{M}^\top)) - \boldsymbol{I}\|^2 && (52)\\
&= \|(\boldsymbol{M}\boldsymbol{M}^\top - \boldsymbol{I}) + 2\lambda\boldsymbol{M}\boldsymbol{M}^\top(\boldsymbol{I} - \boldsymbol{M}\boldsymbol{M}^\top) + \lambda^2\boldsymbol{M}\boldsymbol{M}^\top(\boldsymbol{I} - \boldsymbol{M}\boldsymbol{M}^\top)^2\|^2 && (53)\\
&= \|(\boldsymbol{M}\boldsymbol{M}^\top - \boldsymbol{I}) - 2\lambda\boldsymbol{M}\boldsymbol{M}^\top(\boldsymbol{M}\boldsymbol{M}^\top - \boldsymbol{I}) + \lambda^2\boldsymbol{M}\boldsymbol{M}^\top(\boldsymbol{M}\boldsymbol{M}^\top - \boldsymbol{I})^2\|^2 && (54)\\
&= \|[\boldsymbol{I} - 2\lambda\boldsymbol{M}\boldsymbol{M}^\top + \lambda^2\boldsymbol{M}\boldsymbol{M}^\top(\boldsymbol{M}\boldsymbol{M}^\top - \boldsymbol{I})](\boldsymbol{M}\boldsymbol{M}^\top - \boldsymbol{I})\|^2 \,, && (55)
\end{aligned}$$

where, for $\lambda = 1/2$,

$$\begin{aligned}
\|\boldsymbol{X}_1\boldsymbol{X}_1^\top - \boldsymbol{I}\|^2 &= \|[(\boldsymbol{I} - \boldsymbol{M}\boldsymbol{M}^\top) + \frac{1}{4}\boldsymbol{M}\boldsymbol{M}^\top(\boldsymbol{M}\boldsymbol{M}^\top - \boldsymbol{I})](\boldsymbol{M}\boldsymbol{M}^\top - \boldsymbol{I})\|^2 && (56)\\
&= \|(\boldsymbol{I} - \frac{1}{4}\boldsymbol{M}\boldsymbol{M}^\top)(\boldsymbol{M}\boldsymbol{M}^\top - \boldsymbol{I})^2\|^2 && (57)\\
&\leq \|\boldsymbol{I} - \frac{1}{4}\boldsymbol{M}\boldsymbol{M}^\top\|^2 \|(\boldsymbol{M}\boldsymbol{M}^\top - \boldsymbol{I})^2\|^2 && (58)\\
&\leq \left(\frac{3}{4} + \frac{1}{4}\xi^2\right)^2 \|\boldsymbol{M}\boldsymbol{M}^\top - \boldsymbol{I}\|^4 && (59)\\
&\leq \left(\frac{3}{4} + \frac{1}{4}\xi^2\right)^2 \xi^8 = \mathrm{o}(\xi^7) \quad \text{as } \xi \to 0 \,. && (60)
\end{aligned}$$

□

A.5. Proof of Thm. 3.4

Theorem A.8. *Let $\boldsymbol{X} \in \mathrm{St}^\varepsilon(p,n)$, $\boldsymbol{M} = \boldsymbol{X} - \eta\boldsymbol{X}\boldsymbol{S}$ and $\boldsymbol{X}_1 = \boldsymbol{M} - \lambda(\boldsymbol{M}\boldsymbol{M}^\top - \boldsymbol{I})\boldsymbol{M}$ as defined in Eqs. 10 and 11. If we assume also that $\|\boldsymbol{G}\| \leq L$ and $\xi \coloneqq \eta L < 1$, then*

$$\|\boldsymbol{X}_1\boldsymbol{X}_1^\top - \boldsymbol{I}\|^2 \leq 2P_{\boldsymbol{Y}}(\lambda) + 2[2 + 2\sqrt{P_{\boldsymbol{Y}}(\lambda)} + Q(\lambda,\varepsilon)]^2 Q(\lambda,\varepsilon)^2 \tag{61}$$

where $P_{\boldsymbol{Y}}(\lambda)$ is the landing polynomial of the projection of $\boldsymbol{X}$ to the manifold, $\boldsymbol{Y} = \boldsymbol{U}\boldsymbol{V}^\top$ where $\boldsymbol{X} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top$ is the singular value decomposition of $\boldsymbol{X}$, and where

$$Q(\lambda, \varepsilon) \coloneqq (24\lambda + 3)\varepsilon + \mathrm{o}(\varepsilon) \quad \textit{as } \varepsilon \to 0\,. \tag{62}$$

Proof. Since $\boldsymbol{X} \in \mathrm{St}^\varepsilon(p, n)$, we can use Prop. A.2 and write $\boldsymbol{X} = \boldsymbol{U}(\boldsymbol{I} + \boldsymbol{\Delta})\boldsymbol{V}^\top$ where $\|\boldsymbol{\Delta}\| \leq \varepsilon$. Then, if $\boldsymbol{Y} \coloneqq \boldsymbol{U}\boldsymbol{V}^\top$ is the projection of $\boldsymbol{X}$ onto the Stiefel manifold, $\boldsymbol{X} = \boldsymbol{U}(\boldsymbol{I} + \boldsymbol{\Delta})\boldsymbol{V}^\top = \boldsymbol{Y} + \boldsymbol{U}\boldsymbol{\Delta}\boldsymbol{V}^\top = \boldsymbol{Y} + \boldsymbol{B}$ where we have that $\|\boldsymbol{Y}\| = 1$ and $\|\boldsymbol{B}\| \leq \varepsilon$.

Let us write the Riemannian gradient of $\boldsymbol{X}$ as a function of $\boldsymbol{Y}$ and $\boldsymbol{B}$:

$$\begin{aligned}
\boldsymbol{X}\boldsymbol{S} &= \boldsymbol{X}\,\mathrm{Skew}(\boldsymbol{X}^\top\boldsymbol{G}) && (63)\\
&= (\boldsymbol{Y} + \boldsymbol{B})\,\mathrm{Skew}((\boldsymbol{Y} + \boldsymbol{B})^\top\boldsymbol{G}) && (64)\\
&= (\boldsymbol{Y} + \boldsymbol{B})(\mathrm{Skew}(\boldsymbol{Y}^\top\boldsymbol{G}) + \mathrm{Skew}(\boldsymbol{B}^\top\boldsymbol{G})) && (65)\\
&= (\boldsymbol{Y} + \boldsymbol{B})\,\mathrm{Skew}(\boldsymbol{Y}^\top\boldsymbol{G}) + (\boldsymbol{Y} + \boldsymbol{B})\,\mathrm{Skew}(\boldsymbol{B}^\top\boldsymbol{G}) && (66)\\
&= \boldsymbol{Y}\,\mathrm{Skew}(\boldsymbol{Y}^\top\boldsymbol{G}) + \boldsymbol{B}\,\mathrm{Skew}(\boldsymbol{Y}^\top\boldsymbol{G}) + \boldsymbol{Y}\,\mathrm{Skew}(\boldsymbol{B}^\top\boldsymbol{G}) + \boldsymbol{B}\,\mathrm{Skew}(\boldsymbol{B}^\top\boldsymbol{G}) && (67)\\
&= \boldsymbol{Y}\boldsymbol{S}_{\boldsymbol{Y}} + \boldsymbol{T}_{\boldsymbol{R}}\,, && (68)
\end{aligned}$$

and thus we can write the Riemannian gradient of $\boldsymbol{X}$ as the sum of the Riemannian gradient of $\boldsymbol{Y}$ plus a remainder term.

We can also bound each of these terms:

$$\begin{aligned}
&\|\boldsymbol{B}\,\mathrm{Skew}(\boldsymbol{Y}^\top\boldsymbol{G})\| \leq \|\boldsymbol{B}\|\|\mathrm{Skew}(\boldsymbol{Y}^\top\boldsymbol{G})\| \leq \|\boldsymbol{B}\|\|\boldsymbol{Y}^\top\boldsymbol{G}\| = \|\boldsymbol{B}\|\|\boldsymbol{G}\| \leq L\varepsilon && (69)\\
&\|\boldsymbol{Y}\,\mathrm{Skew}(\boldsymbol{B}^\top\boldsymbol{G})\| \leq \|\boldsymbol{Y}\|\|\boldsymbol{B}^\top\boldsymbol{G}\| = \|\boldsymbol{B}^\top\boldsymbol{G}\| \leq \|\boldsymbol{B}\|\|\boldsymbol{G}\| \leq L\varepsilon && (70)\\
&\|\boldsymbol{B}\,\mathrm{Skew}(\boldsymbol{B}^\top\boldsymbol{G})\| \leq \|\boldsymbol{B}\|\|\boldsymbol{B}^\top\boldsymbol{G}\| \leq \|\boldsymbol{B}\|^2\|\boldsymbol{G}\| \leq L\varepsilon^2 && (71)\\
&\|\boldsymbol{T}_{\boldsymbol{R}}\| \leq 2L\varepsilon + L\varepsilon^2 = (2 + \varepsilon)L\varepsilon && (72)
\end{aligned}$$

Now we can do something similar with $\boldsymbol{M}$;

$$\begin{aligned}
\boldsymbol{M} &= \boldsymbol{X} - \eta\boldsymbol{X}\boldsymbol{S} && (73)\\
&= \boldsymbol{Y} + \boldsymbol{B} - \eta\boldsymbol{Y}\boldsymbol{S}_{\boldsymbol{Y}} - \eta\boldsymbol{T}_{\boldsymbol{R}} && (74)\\
&= \boldsymbol{Y}_1 + \boldsymbol{B} - \eta\boldsymbol{T}_{\boldsymbol{R}} && (75)\\
&= \boldsymbol{Y}_1 + \boldsymbol{T}_1\,, && (76)
\end{aligned}$$

and bound each term:

$$\begin{aligned}
\|\boldsymbol{Y}_1\| = \|\boldsymbol{Y} - \eta\boldsymbol{Y}\boldsymbol{S}_{\boldsymbol{Y}}\| &\leq \|\boldsymbol{Y}\| + \eta\|\boldsymbol{Y}\boldsymbol{S}_{\boldsymbol{Y}}\| = 1 + \eta\|\boldsymbol{Y}\boldsymbol{S}_{\boldsymbol{Y}}\| && (77)\\
&= 1 + \eta\|\boldsymbol{S}_{\boldsymbol{Y}}\| \leq 1 + \eta\|\boldsymbol{G}\| \leq 1 + \eta L = 1 + \xi < 2 && (78)
\end{aligned}$$

$$\begin{aligned}
\|\boldsymbol{T}_1\| = \|\boldsymbol{B} - \eta\boldsymbol{T}_{\boldsymbol{R}}\| &\leq \|\boldsymbol{B}\| + \|-\eta\boldsymbol{T}_{\boldsymbol{R}}\| = \|\boldsymbol{B}\| + \eta\|\boldsymbol{T}_{\boldsymbol{R}}\| && (79)\\
&\leq \varepsilon + \eta(2 + \varepsilon)L\varepsilon = \varepsilon + (2 + \varepsilon)\varepsilon\xi < (3 + \varepsilon)\varepsilon && (80)
\end{aligned}$$

We can once again repeat the same process for the distance gradient:

$$\begin{aligned}
\nabla\mathcal{N}(\boldsymbol{M}) &= (\boldsymbol{M}\boldsymbol{M}^\top - \boldsymbol{I})\boldsymbol{M} && (81)\\
&= ((\boldsymbol{Y}_1 + \boldsymbol{T}_1)(\boldsymbol{Y}_1^\top + \boldsymbol{T}_1^\top) - \boldsymbol{I})\boldsymbol{M} && (82)\\
&= (\boldsymbol{Y}_1(\boldsymbol{Y}_1^\top + \boldsymbol{T}_1^\top) + \boldsymbol{T}_1(\boldsymbol{Y}_1^\top + \boldsymbol{T}_1^\top) - \boldsymbol{I})\boldsymbol{M} && (83)\\
&= (\boldsymbol{Y}_1\boldsymbol{Y}_1^\top + \boldsymbol{Y}_1\boldsymbol{T}_1^\top + \boldsymbol{T}_1\boldsymbol{Y}_1^\top + \boldsymbol{T}_1\boldsymbol{T}_1^\top - \boldsymbol{I})\boldsymbol{M} && (84)\\
&= (\boldsymbol{Y}_1\boldsymbol{Y}_1^\top + \boldsymbol{Y}_1\boldsymbol{T}_1^\top + \boldsymbol{T}_1\boldsymbol{Y}_1^\top + \boldsymbol{T}_1\boldsymbol{T}_1^\top - \boldsymbol{I})(\boldsymbol{Y}_1 + \boldsymbol{T}_1) && (85)\\
&= (\boldsymbol{Y}_1\boldsymbol{Y}_1^\top + \boldsymbol{Y}_1\boldsymbol{T}_1^\top + \boldsymbol{T}_1\boldsymbol{Y}_1^\top + \boldsymbol{T}_1\boldsymbol{T}_1^\top - \boldsymbol{I})\boldsymbol{Y}_1 + (\boldsymbol{M}\boldsymbol{M}^\top - \boldsymbol{I})\boldsymbol{T}_1 && (86)
\end{aligned}$$

$$= \nabla \mathcal{N}(\boldsymbol{Y}_1) + (\boldsymbol{Y}_1\boldsymbol{T}_1^\top + \boldsymbol{T}_1\boldsymbol{Y}_1^\top + \boldsymbol{T}_1\boldsymbol{T}_1^\top)\boldsymbol{Y}_1 + (\boldsymbol{M}\boldsymbol{M}^\top - \boldsymbol{I})\boldsymbol{T}_1 \tag{87}$$
$$= \nabla \mathcal{N}(\boldsymbol{Y}_1) + \boldsymbol{T}_\mathcal{N} \tag{88}$$

and bound each term:

$$\|(\boldsymbol{Y}_1\boldsymbol{T}_1^\top + \boldsymbol{T}_1\boldsymbol{Y}_1^\top)\boldsymbol{Y}_1\| \le \|2\,\mathrm{Sym}(\boldsymbol{Y}_1\boldsymbol{T}_1^\top)\boldsymbol{Y}_1\| \le 2\|\boldsymbol{Y}_1\boldsymbol{T}_1^\top\|\|\boldsymbol{Y}_1\| \le 2\|\boldsymbol{Y}_1\|\|\boldsymbol{T}_1^\top\|\|\boldsymbol{Y}_1\| \tag{89}$$
$$= 2(1+\xi)^2(\varepsilon + (2+\varepsilon)\varepsilon\xi) < 8(3+\varepsilon)\varepsilon \tag{90}$$

$$\|\boldsymbol{T}_1\boldsymbol{T}_1^\top\boldsymbol{Y}_1\| \le \|\boldsymbol{T}_1\|\|\boldsymbol{T}_1^\top\|\|\boldsymbol{Y}_1\| = (1+\xi)(\varepsilon + (2+\varepsilon)\varepsilon\xi)^2 < 2(3+\varepsilon)^2\varepsilon^2 \tag{91}$$

$$\|\boldsymbol{M}\boldsymbol{M}^\top - \boldsymbol{I}\| = \|(\boldsymbol{X} - \eta\boldsymbol{X}\boldsymbol{S})(\boldsymbol{X} - \eta\boldsymbol{X}\boldsymbol{S})^\top - \boldsymbol{I})\| \tag{92}$$
$$\le \|(\boldsymbol{X}\boldsymbol{X}^\top - \boldsymbol{I}) - \eta(\boldsymbol{X}\boldsymbol{S}^\top\boldsymbol{X}^\top + \boldsymbol{X}\boldsymbol{S}\boldsymbol{X}^\top) + \eta^2\boldsymbol{X}\boldsymbol{S}\boldsymbol{S}^\top\boldsymbol{X}^\top\| \tag{93}$$
$$\le \|(\boldsymbol{X}\boldsymbol{X}^\top - \boldsymbol{I})\| + \eta^2\|\boldsymbol{X}\boldsymbol{S}\boldsymbol{S}^\top\boldsymbol{X}^\top\| \tag{94}$$
$$\le \varepsilon + \eta^2\|\boldsymbol{X}\boldsymbol{S}\|^2 \tag{95}$$
$$\le \varepsilon + \eta^2\|\boldsymbol{X}\|^2\|\boldsymbol{X}^\top\boldsymbol{G}\|^2 \tag{96}$$
$$\le \varepsilon + \xi^2(1+\varepsilon)^2 \tag{97}$$
$$< \varepsilon + (1+\varepsilon)^2 \tag{98}$$

$$\|\boldsymbol{T}_\mathcal{N}\| < 8(3+\varepsilon)\varepsilon + 2(3+\varepsilon)^2\varepsilon^2 + [\varepsilon + (1+\varepsilon)^2](3+\varepsilon)\varepsilon \tag{99}$$
$$= 24\varepsilon + \mathrm{o}(\varepsilon) \quad \text{as } \varepsilon \to 0 \tag{100}$$

Finally, we do the same bounding to our quantity of interest:

$$\boldsymbol{X}_1 = \boldsymbol{M} - \lambda(\boldsymbol{M}\boldsymbol{M}^\top - \boldsymbol{I})\boldsymbol{M} \tag{101}$$
$$= \boldsymbol{Y}_1 + \boldsymbol{T}_1 - \lambda\nabla\mathcal{N}(\boldsymbol{Y}_1) - \lambda\boldsymbol{T}_\mathcal{N} \tag{102}$$
$$= \boldsymbol{Y}_2 + \boldsymbol{T}_1 - \lambda\boldsymbol{T}_\mathcal{N} \tag{103}$$
$$= \boldsymbol{Y}_2 + \boldsymbol{T}_2 \tag{104}$$

$$\|\boldsymbol{T}_2\| \le \|\boldsymbol{T}_1\| + \lambda\|\boldsymbol{T}_\mathcal{N}\| \tag{105}$$
$$< (3+\varepsilon)\varepsilon + \lambda(24\varepsilon + \mathrm{o}(\varepsilon)) \tag{106}$$
$$= (24\lambda + 3)\varepsilon + \mathrm{o}(\varepsilon) =: Q(\lambda, \varepsilon) \tag{107}$$

$$\|\boldsymbol{X}_1\boldsymbol{X}_1^\top - \boldsymbol{I}\|^2 = \|(\boldsymbol{Y}_2 + \boldsymbol{T}_2)(\boldsymbol{Y}_2 + \boldsymbol{T}_2)^\top - \boldsymbol{I}\|^2 \tag{108}$$
$$= \|\boldsymbol{Y}_2(\boldsymbol{Y}_2^\top + \boldsymbol{T}_2^\top) + \boldsymbol{T}_2(\boldsymbol{Y}_2^\top + \boldsymbol{T}_2^\top) - \boldsymbol{I}\|^2 \tag{109}$$
$$= \|\boldsymbol{Y}_2\boldsymbol{Y}_2^\top + \boldsymbol{Y}_2\boldsymbol{T}_2^\top + \boldsymbol{T}_2(\boldsymbol{Y}_2^\top + \boldsymbol{T}_2^\top) - \boldsymbol{I}\|^2 \tag{110}$$
$$= \|\boldsymbol{Y}_2\boldsymbol{Y}_2^\top - \boldsymbol{I} + \boldsymbol{Y}_2\boldsymbol{T}_2^\top + \boldsymbol{T}_2(\boldsymbol{Y}_2^\top + \boldsymbol{T}_2^\top)\|^2 \tag{111}$$
$$\le 2\|\boldsymbol{Y}_2\boldsymbol{Y}_2^\top - \boldsymbol{I}\|^2 + 2\|\boldsymbol{Y}_2\boldsymbol{T}_2^\top + \boldsymbol{T}_2(\boldsymbol{Y}_2^\top + \boldsymbol{T}_2^\top)\|^2 \tag{112}$$

$$\|\boldsymbol{Y}_2\boldsymbol{T}_2^\top + \boldsymbol{T}_2(\boldsymbol{Y}_2^\top + \boldsymbol{T}_2^\top)\|^2 = \|2\,\mathrm{Sym}(\boldsymbol{Y}_2\boldsymbol{T}_2^\top) + \boldsymbol{T}_2\boldsymbol{T}_2^\top\|^2 \tag{113}$$
$$\le (\|2\,\mathrm{Sym}(\boldsymbol{Y}_2\boldsymbol{T}_2^\top)\| + \|\boldsymbol{T}_2\boldsymbol{T}_2^\top\|)^2 \tag{114}$$
$$\le (2\|\boldsymbol{Y}_2\|\|\boldsymbol{T}_2\| + \|\boldsymbol{T}_2\|^2)^2 \tag{115}$$
$$= (2\|\boldsymbol{Y}_2\| + \|\boldsymbol{T}_2\|)^2\|\boldsymbol{T}_2\|^2 \tag{116}$$

$$\leq [2(1+\sqrt{P_{\boldsymbol{Y}}(\lambda)})+Q(\lambda,\varepsilon)]^2 Q(\lambda,\varepsilon)^2 \tag{117}$$

And therefore,

$$\|\boldsymbol{X}_1\boldsymbol{X}_1^\top - \boldsymbol{I}\|^2 \leq 2\|\boldsymbol{Y}_2\boldsymbol{Y}_2^\top - \boldsymbol{I}\|^2 + 2\|\boldsymbol{Y}_2\boldsymbol{T}_2^\top + \boldsymbol{T}_2(\boldsymbol{Y}_2^\top + \boldsymbol{T}_2^\top)\|^2 \tag{118}$$

$$\leq 2P_{\boldsymbol{Y}}(\lambda) + 2[2+2\sqrt{P_{\boldsymbol{Y}}(\lambda)}+Q(\lambda,\varepsilon)]^2 Q(\lambda,\varepsilon)^2 \tag{119}$$

□

A.6. Proof of Thm. 3.5

Theorem A.9. *If POGO is initialized with $\boldsymbol{X}_0 \in \mathrm{St}(p,n)$, $\lambda = 1/2$ and $\xi := \eta L < 1$, then we have that*

$$\|\boldsymbol{X}_t\boldsymbol{X}_t^\top - \boldsymbol{I}\|^2 = \mathrm{o}(\xi^7) \quad \textit{for every iteration } t \in \mathbb{N}\,. \tag{120}$$

Proof. Say that we start at $\boldsymbol{X}_0 \in \mathrm{St}(p,n)$ and that $\xi < 1$. Then, $\|\boldsymbol{X}_1\boldsymbol{X}_1^\top - \boldsymbol{I}\|^2 = 4P(1/2) = \mathrm{o}(\xi^7)$ via Prop. A.7. So $\boldsymbol{X}_1 \in \mathrm{St}^\varepsilon(p,n)$ with $\varepsilon = \mathrm{o}(\xi^{7/2})$. Then, we have for the next iteration $\boldsymbol{X}_2$ that

$$Q(1/2, \mathrm{o}(\xi^3)) = (12+3)\,\mathrm{o}(\xi^{7/2}) + \mathrm{o}(\mathrm{o}(\xi^{7/2})) = \mathrm{o}(\xi^{7/2}) \tag{121}$$

and thus

$$\|\boldsymbol{X}_2\boldsymbol{X}_2^\top - \boldsymbol{I}\|^2 \leq 2\,\mathrm{o}(\xi^7) + 2[2+2\sqrt{\mathrm{o}(\xi^7)}+\mathrm{o}(\xi^{7/2})]^2\,\mathrm{o}(\xi^{7/2})^2 \tag{122}$$

$$= \mathrm{o}(\xi^7) + 2[2+\mathrm{o}(\xi^{7/2})+\mathrm{o}(\xi^{7/2})]^2\,\mathrm{o}(\xi^7) \tag{123}$$

$$= \mathrm{o}(\xi^7)\,. \tag{124}$$

Since ε remains $\mathrm{o}(\xi^{7/2})$ and the rest of parameters are kept fixed as well ($\lambda = 1/2$, η and L), then we prove by induction that

$$\|\boldsymbol{X}_t\boldsymbol{X}_t^\top - \boldsymbol{I}\|^2 = \mathrm{o}(\xi^7)\,, \tag{125}$$

where t is the iteration number, i.e., all the iterations of POGO remain $\mathrm{o}(\xi^{7/2})$-close to the manifold as long as $\xi = \eta L < 1$.

□

B. VectorAdam as a Linear Optimizer

In this section, we describe why VectorAdam (VAdam) (Ling et al., 2022) meets the definition of a linear optimizer (Def. 1). First, VAdam computes the vector update as a ratio $m_{t,i}/v_{t,i}$, where the numerator is an exponential moving average (EMA) of the gradient, i.e.:

$$m_t = \mathrm{EMA}_t(\nabla f(\boldsymbol{X}_t); \beta_1) = \beta_1 m_{t-1} + (1-\beta_1)\nabla f(\boldsymbol{X}_t). \tag{126}$$

And then it it easy to show that the numerator is equivariant w.r.t. the relative gradient (recall, $\mathrm{Skew}(\boldsymbol{A}) = \frac{1}{2}(\boldsymbol{A} - \boldsymbol{A}^\top)$):

$$\boldsymbol{S}_{\text{VAdam}} = \boldsymbol{X}_t\,\mathrm{Skew}(\boldsymbol{X}_t^\top \mathrm{EMA}_t(\nabla f(\boldsymbol{X}_t); \beta_1)) \tag{127}$$

$$= \boldsymbol{X}_t\,\mathrm{Skew}(\boldsymbol{X}_t^\top m_t) = \boldsymbol{X}_t \frac{1}{2}(\boldsymbol{X}_t^\top m_t - m_t^\top \boldsymbol{X}_t) \tag{128}$$

$$= \beta_1 \boldsymbol{X}_t\,\mathrm{Skew}(\boldsymbol{X}_t m_{t-1} x) + (1-\beta_1)\boldsymbol{X}_t\,\mathrm{Skew}(\boldsymbol{X}_t^\top \nabla f(\boldsymbol{X}_t)) = \tag{129}$$

$$= \mathrm{EMA}_t(\boldsymbol{X}_t\,\mathrm{Skew}(\boldsymbol{X}_t^\top \nabla f(\boldsymbol{X}_t)); \beta_1) = \mathrm{EMA}_t(\boldsymbol{S}_{\nabla f(\boldsymbol{X}_t)}) \tag{130}$$

That is, since the relative gradient operator is linear, it is equivalent to compute the numerator of VAdam (in the Euclidean space) and then its relative gradient, than compute the numerator of VAdam for the relative gradient (i.e. in the Stiefel manifold). Finally, the denominator of VAdam is a scalar shared for all parameters, i.e., $v_{t,i} = v_t = ||\nabla f(\boldsymbol{X}_t)||^2$. Hence, VAdam is a linear optimizer as in Def. 1.

C. Relation of POGO with SLPG

In this section, we briefly present the original formulation of SLPG (Liu et al., 2024) as well as under which conditions and the derivations needed to recover something similar to the updates of POGO.

First, note that SLPG was initially introduced by Liu et al. (2024) in the context of optimization problems with orthogonality constraints and a non-smooth regularization term. Also note that the following formulas use the original formulation by Liu et al. (2024) which consider *column-orthogonal* matrices, such that $\boldsymbol{X}^\top \boldsymbol{X} = \boldsymbol{I}_p$. Within this context, Liu et al. (2024) attempt to solve the following problem:

$$\min_{\boldsymbol{X}\in\mathbb{R}^{n\times p}} f(\boldsymbol{X}) + r(\boldsymbol{X}) \quad \text{s.t.} \quad \boldsymbol{X}^\top \boldsymbol{X} = \boldsymbol{I}_p\,. \tag{131}$$

And make the following assumptions (we disregard those from r as they are of no use in our context):

1. f is differentiable and locally L-smooth.
2. For any $\boldsymbol{X}, \boldsymbol{G} \in \mathbb{R}^{n\times p}$ the problem

$$\min_{\boldsymbol{D}\in\mathbb{R}^{n\times p}} \langle \boldsymbol{G}, \boldsymbol{D}\rangle + r(\boldsymbol{D}) + \frac{1}{2\eta}\|\boldsymbol{D}-\boldsymbol{X}\|^2 \tag{132}$$

   has a closed-form solution that can be efficiently solved by certain iterative approach.

Then, Liu et al. (2024) propose an iterative process that consist in the following steps:

1. While the following terminating condition is not met:

$$\|\mathrm{Sym}((\boldsymbol{Y}_k - \boldsymbol{X}_k)^\top \boldsymbol{X}_k)\| \le c\|\boldsymbol{X}^\top\boldsymbol{X} - \boldsymbol{I}\|\,, \tag{133}$$

2. Approximately solve the following proximal optimization problem:

$$\min_{\boldsymbol{D}\in\mathbb{R}^{n\times p}} \langle \nabla f(\boldsymbol{X}), \boldsymbol{D}\rangle + r(\boldsymbol{D}) + \frac{1}{2\eta}\|\boldsymbol{D}-\boldsymbol{X}_k\|^2 \tag{134}$$

   with the following iterative process until convergence:

   (a) Calculate the proximal mapping

$$\boldsymbol{D}_j = \mathrm{prox}_\eta(\nabla f(\boldsymbol{X}_k) - \boldsymbol{X}_k\Lambda_j; \boldsymbol{X}_k) = \operatorname*{arg\,min}_{\boldsymbol{D}\in\mathbb{R}^{n\times p}} \langle \boldsymbol{G}, \boldsymbol{D}\rangle + r(\boldsymbol{D}) + \frac{1}{2\eta}\|\boldsymbol{D}-\boldsymbol{X}\|^2 \tag{135}$$

   (b) Update $\Lambda_{j+1} = \Lambda_j - \frac{1}{\eta}\,\mathrm{Sym}((\boldsymbol{D}_j - \boldsymbol{X}_k)^\top \boldsymbol{X}_k)$
3. Set $\boldsymbol{Y}_k = \boldsymbol{D}_J$.
4. Compute an approximate normal step, for which they use a first-order Taylor approximation of a polar retraction, $\boldsymbol{X}_{k+1} = \boldsymbol{Y}_k(\frac{3}{2}\boldsymbol{I}_p - \frac{1}{2}\boldsymbol{Y}_k^\top\boldsymbol{Y}_k)$.

Relationship with POGO. As discussed in the main text, the last step in the algorithm above corresponds to the same normal update as POGO for the case when we set $\lambda = 1/2$, despite coming from completely different angles. Moreover, if we assume that we have a smooth problem $r = 0$, the authors point out that then the Lagrangian Λ has a explicit form, $\Lambda(\boldsymbol{X}) = \mathrm{Sym}(\boldsymbol{X}^\top\nabla f(\boldsymbol{X}))$. Now, if we look at the proximal problem for the case when $r = 0$, it turns out that it looks like the following:

$$\min_{\boldsymbol{D}\in\mathbb{R}^{n\times p}} \langle \boldsymbol{G}, \boldsymbol{D}\rangle + \frac{1}{2\eta}\|\boldsymbol{D}-\boldsymbol{X}\|^2 \tag{136}$$

Then, the derivative of the optimization function w.r.t. $\boldsymbol{D}$ is of the form $\boldsymbol{G} + \frac{1}{\eta}(\boldsymbol{D}-\boldsymbol{X})$ which we can see easily that is zero when $\boldsymbol{D} = \boldsymbol{X} - \eta\boldsymbol{G}$, and that is therefore the minimum for the proximal problem for this case. Finally, if we plug Λ inside the solution of the proximal problem we get that $\boldsymbol{Y}_k = \boldsymbol{X}_k - \eta(\nabla f(\boldsymbol{X}) - \boldsymbol{X}_k\,\mathrm{Sym}(\boldsymbol{X}^\top\nabla f(\boldsymbol{X})))$.

Therefore, if we set $\lambda = 1/2$ and *not use any base optimizer*, we recover the normal step from POGO as well as a similar intermediate step, the difference being that we use $\boldsymbol{X}\,\mathrm{Skew}(\boldsymbol{X}^\top\nabla f(\boldsymbol{X}))$, the Riemannian gradient under the canonical metric, and SLPG uses $\nabla f(\boldsymbol{X}) - \boldsymbol{X}_k\,\mathrm{Sym}(\boldsymbol{X}^\top\nabla f(\boldsymbol{X}))$, the Riemannian gradient under the Euclidean metric. Both are

quite similar, and indeed coincide when $p = 1$ or $p = n$. The main difference otherwise is that the one that POGO uses is orthogonal to the normal direction, while the one used by SLPG contains an extra component which can drift the gradient update outside the tangent space.

Despite the similarities in methodology, it is worth noting that both papers start from quite different premises and make a different number of assumptions and theoretical contributions. For example, in this work we do not consider non-smooth regularization, but we provide a tighter bound for the distance of the updates, milder assumptions on the learning rate, as well as the introduction of unconstrained optimizers into the orthoptimizer. To the best of our knowledge, we are the first to implement SLPG besides the original authors, and the first to derive the simplified version of SLPG for which we can recover POGO for certain cases.

D. Experimental Details and Results

In this section we provide additional experimental details in order to reproduce the experiments from the main section. All the experiments in the paper were produced under the same condition using a small cluster with 8 NVIDIA RTX A6000 GPUs. We will make sure to release our code to easily reproduce our experiments upon acceptance.

For every plot showing time we use linear interpolation to sample at the same time steps for every independent run, taking then the average and, when it did not clutter too much the image as in Fig. 4, plot a confidence interval of 90 for performances and 70 for distances (due to the log-scale producing artifacts). Unless otherwise specified, we use the default hyperparameter for all methods. Except for the PCA and Procrustes experiments (which we explain below), hyperparameters were search with a grid search of equal budget for all baselines, picking the set of hyperparameters that obtained best validation performance.

D.1. Online PCA and Orthogonal Procrustes

For these two experiments we adopted the codebase of RSDM which is publicly available at https://github.com/andyjm3/RSDM. Then, we simply had to introduce each of the baselines as drop-in replacements for the previously used optimizers. We additionally modified the code to log distances at each step, as well as to fix the randomness for each experiment. Given that running each of these experiments is a matter of a few seconds, we manually tuned the hyperparameters of each individual orthoptimizer until we could not get any better results.

In particular, we used for PCA $p = 1500$, $n = 2000$, set the dimension of RSDM to 700 and then set the learning rate of each method as follows: 0.15 for RGD, 1.5 for RSDM, 0.25 for Landing and POGO, 10.5 for LandingPC, and 0.125 for SLPG. We used SGD as base optimizer for POGO with momentum of 0.3, set the momentum for Landing to 0.1 and the λ parameter of LandingPC to 0.01 (for Landing we kept the default value of 1). Then, we trained with each method 10 independent times and aggregated the results to take the average, stopping if we reached an optimality gap of 1×10^{-6}.

Similarly, for Procrustes we set $n = p = 2000$ with a dimension for RSDM of 900. Then we employed the following learning rates: 0.5 for RGD, 2 for RSDM, 0.5 for Landing and POGO, 1.5 for LandingPC, and 0.5 for SLPG. For LandingPC we set λ to 0.1 this time, and reduced the momentum for the SGD base optimizer of POGO to 0.1, sharing the same momentum as for Landing.

D.2. Vision Transfomer

Similar to PCA and Procrustes, we adapted the ViT experiment from the repository of RSDM located at https://github.com/andyjm3/RSDM. In this case, we ran every experiment five independent times and trained for 10 epochs. We kept the dimension of 300 for the submanifold of RSDM as in the original code, and use the following learning rates after performing a grid search: 0.1 for RGD, 0.5 for RSDM, 0.001 for Landing, 0.01 for LandingPC and SLPG, as well as for POGO which also used SGD as base optimizer. Additionally, we set the momentum of Landing to 0.1.

D.3. Convolutional Neural Network

For the experiments with the CNN, we adapted the code for the CIFAR-10 speedrun benchmarks, which is publicly available here: https://github.com/KellerJordan/cifar10-airbench/. In terms of changes, we had to make sure that the initial parameters were orthogonal, for which we simply projected to the Stiefel manifold at initialization, as well as modified the code to impose the orthogonal constraints to the filters and kernels, depending of the experiment. We increased the default number of epochs from 20 to 100 and repeated all the experiments 5 times.

After a grid search with equal budget, these are the hyperparameters that we used:

1. Orthogonal filters: learning rates of 0.01 for RGD and Adam, 0.1 for RSDM (with a submanifold dimension of 64), 0.001 for SLPG and Landing (the latter with a momentum of 0.6), and we used a learning rate of 0.5 for LandingPC and POGO. Additionally, we use VAdam as base optimizer for POGO.
2. Orthogonal kernels: learning rates of 0.01 for RGD, Adam and Landing (this time with no momentum), 0.5 for RSDM (with a submanifold dimension of 2 out of 3), 0.1 for SLPG, and 0.5 again for LandingPC and POGO (using VAdam as base optimizer).

D.4. Squared Unitary Probabilistic Circuits

We consider probabilistic circuits (PCs) (Choi et al., 2020; Vergari et al., 2021) that are representing a complex wave function and then a probability distribution once squared (Loconte et al., 2024; 2025). For our experiments, these squared PCs are parameterized with complex unitary matrices. We could not find any available code online for Loconte et al. (2026). After contacting the authors through email, they provided access to their codebase to reproduce the experiments in their work. Therefore, we refer the readers to the experimental details within (Loconte et al., 2026) for details on how to reproduce those experiments, as we did not change anything other than plugging in our baseline orthoptimizers.

In particular, the MNIST experiment we show in this article corresponds to the squared unitary probabilistic circuit with Kronecker products and 10 units. The only difference with respect to their experimental setting is the halving of learning rate after finding a plateau, which we noticed it improved the results of all baselines.

Regarding the hyperparameter used for the optimizers, these are the values that we found to work the best given an equal budget between methods: learning rate of 0.05 for RGD and LandingPC (the latter with a value λ of 0.1 in addition), 0.01 for Landing and 0.5 for POGO using VAdam as base optimizer. With respect to SLPG, we found that our considered learning rates lead to the model diverging within the first epoch, and so we found the best learning rate to be 0.0005 (0.0075 already diverged). In a similar note, we could not make work RSDM with a submanifold size of 8 for every value of learning rate that we tried: While the method did run, its performance was subpar to the rest of methods, not even lying within the range of the plots shown in the main paper.

D.5. Ablation on Tensor Precision

In order to understand the reason why RSDM was monotonically increasing the distance of its iterates from the manifold, we decided to run a small experiment on the online PCA setting.

In this experiment, we took a subset of orthoptimizers to have as a reference for RSDM, and then reproduce the experiments from the main paper with two different settings. In the first, we activate the use of 16-bit precision floating points to compute matrix multiplications (still storing them as 32-bit precision numbers). This option sacrifices accuracy in the matrix multiplication by significant gains in running times. The second experiment instead ran all the experiment with 64-bit precision floating points, which significantly slowed the experiments in exchange of more accurate computations. It is important to remark that to track the results (optimality gap and manifold distance) we went back to the original setting at each iteration.

The results are summarized in in Fig. D.1, where we can make a couple of observations. Regarding training time, we see that POGO and Landing benefit the most from speed-ups in matrix multiplication and, conversely, all methods significantly increased training times (from 20 to 500 seconds) when using 64-bit floating point precision. Regarding manifold distances, we first observe that all considered methods converge to the manifold for the highest precision, *including RSDM*. This points to numerical problems within the computations of RSDM. Similarly, we see that the use of faster matrix multiplication come at the cost of less precision on the feasibility of the solutions, with all distances increasing order several orders of magnitude with respect to those results from Fig. 4.

D.6. Ablation on Computing the Step Size

In order to clearly show the advantage that POGO has from leveraging unconstrained optimizers, as well as to empirically demonstrate the theoretical results discussed in §§3.2 and 3.3, we ran an ablation study on this squared unitary PC experiment (§5.3), where we take POGO with no base optimizer ($\boldsymbol{G} = \nabla f(\boldsymbol{X})$) and vary the learning rate for $\eta \in \{0.001, 0.005, 0.007, 0.010, 0.025\}$ for the case where we compute λ by solving the landing polynomial (see §3.2) or by setting λ to $1/2$.

We present the results for each of these two options in Fig. D.2, where we also plot for reference POGO with VAdam as base optimizer (the one selected for the experiments in §5.3). First, we observe that the version with VAdam obtains the best bpd results than any of the other hyperparameter settings, showing the competitive advantage that unconstrained optimizers can

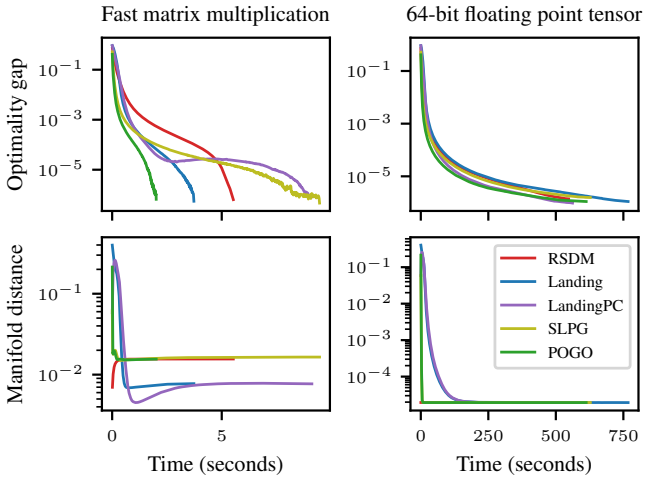

Figure D.1. Ablation on online PCA with a subset of methods exploring the trade-off between multiplication speed and numerical precision.

provide to POGO. Second, we see that when we compute λ and increase the learning rate, POGO goes from staying in the manifold all the time to fluctuate more, to the point where it directly escapes from the manifold at $\eta = 0.025$. For the case where we fix λ to 0.5, we see that only two learning rates appear in Fig. D.2 (right). The reason is that every other version not appearing in the plot diverged within the first epoch, showing that $\lambda = 0.5$ is indeed an approximation that works when we keep ξ under control, and also that if we compute λ we can use higher learning rates.

Finally, we show in Fig. D.3 the same models, where we put together the four versions of POGO with SGD sharing learning rate. In this figure, we can observe that there is no difference at all (besides training time) between fixing λ or computing the root for the smallest learning rate (although the downstream performance is worse). We see a similar trend for $\lambda = 0.005$ for the downstream performance, where both versions of POGO are indistinguishable. However, we observe that in the distance to the manifold fixing λ seems to converge faster. We foresee two possible explanations. First, it could be that our choice for picking the root is not the best one. Second, it is possible that at the beginning of training $\lambda = 0.5$ overshoots the correct root, alternating between both orientations of the manifold at each iteration.

However, these results empirically show that we can control how tight POGO stay on the manifold by reducing the learning rate. In practice, and as we show every time we use VAdam, the gradient normalization that it performs internally helps us to adaptively control $\|\boldsymbol{G}\|$, staying always really close to the manifold and allowing for larger learning rates.

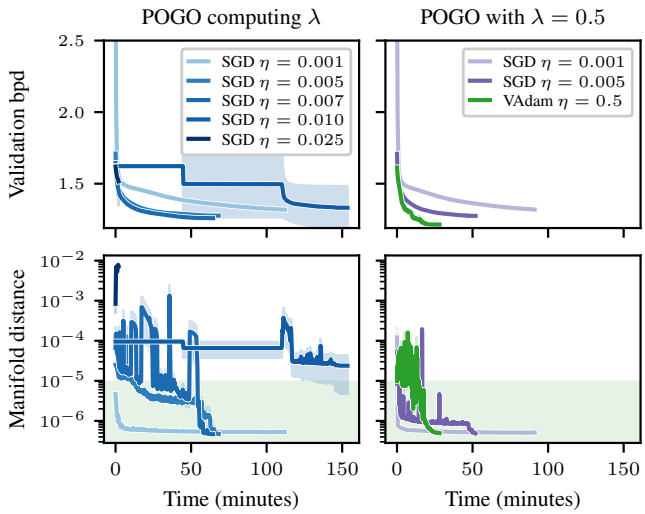

Figure D.2. Ablation of POGO on the PC experiment as we change the learning rate with no base optimizer. Left plots solve the landing polynomial to compute λ and right plots fix λ to $1/2$. POGO with VAdam (Ling et al., 2022) is added as a reference.

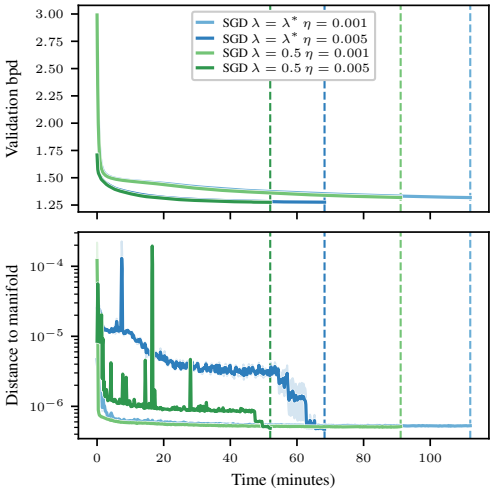

Figure D.3. Same ablation as in Fig. D.2, plotting together POGO solving the landing and polynomial and POGO with $\lambda = 1/2$ for $\eta = 0.001$ and $\eta = 0.005$. We see that both approaches are identical for the smallest step size, while they differ at the largest one, likely from $\lambda = 1/2$ over/under-estimating the value of λ^*.