Diagonal Acceleration for Covariance Matrix Adaptation Evolution Strategies

Youhei Akimoto, Nikolaus Hansen

Diagonal Acceleration for Covariance Matrix Adaptation Evolution Strategies

Y. Akimoto akimoto@cs.tsukuba.ac.jp
University of Tsukuba, 1-1-1 Tennodai, Tsukuba, JAPAN

N. Hansen nikolaus.lastname@inria.fr
Inria, RandOpt Team, CMAP, Ecole Polytechnique, Palaiseau, FRANCE

Abstract

We introduce an acceleration for covariance matrix adaptation evolution strategies (CMA-ES) by means of *adaptive diagonal decoding* (dd-CMA). This diagonal acceleration endows the default CMA-ES with the advantages of separable CMA-ES without inheriting its drawbacks. Technically, we introduce a diagonal matrix D that expresses coordinate-wise variances of the sampling distribution in DCD form. The diagonal matrix can learn a rescaling of the problem in the coordinates within linear number of function evaluations. Diagonal decoding can also exploit separability of the problem, but, crucially, does not compromise the performance on non-separable problems. The latter is accomplished by modulating the learning rate for the diagonal matrix based on the condition number of the underlying correlation matrix. dd-CMA-ES not only combines the advantages of default and separable CMA-ES, but may achieve overadditive speedup: it improves the performance, and even the scaling, of the better of default and separable CMA-ES on classes of non-separable test functions that reflect, arguably, a landscape feature commonly observed in practice.

The paper makes two further secondary contributions: we introduce two different approaches to guarantee positive definiteness of the covariance matrix with active CMA, which is valuable in particular with large population size; we revise the default parameter setting in CMA-ES, proposing accelerated settings in particular for large dimension.

All our contributions can be viewed as independent improvements of CMA-ES, yet they are also complementary and can be seamlessly combined. In numerical experiments with dd-CMA-ES up to dimension 5120, we observe remarkable improvements over the original covariance matrix adaptation on functions with coordinate-wise ill-conditioning. The improvement is observed also for large population sizes up to about dimension squared.

Keywords

Evolution strategies, covariance matrix adaptation, adaptive diagonal decoding, active covariance matrix update, default strategy parameters.

1 Introduction

In real world applications of continuous optimization involving simulations such as physics or chemical simulations, the input-output relation between a candidate solution and its objective function value is barely expressible in explicit mathematical formula. The objective function value is computed through a complex simulation with a candidate solution as an input. In such scenarios, we gain the information of the problem only through the evaluation of the objective function value of a given candidate

solution. A continuous optimization of $f : \mathbb{R}^n \to \mathbb{R}$ is referred to as black-box continuous optimization if we gain the information of the problem only through the evaluation $\boldsymbol{x} \mapsto f(\boldsymbol{x})$ of a given candidate solution $\boldsymbol{x} \in \mathbb{R}^n$.

Black-box continuous optimization arises widely in real world applications such as model parameter calibration and design of robot controller. It often involves computationally expensive simulation to evaluate the quality of candidate solutions. The search cost of black-box continuous optimization is therefore the number of simulations, i.e., the number of objective function calls, and a search algorithm is desired to locate good quality solutions with as few f-calls as possible. Practitioners need to choose one or a few search algorithms to solve their problems and tune their hyper-parameters based on the prior knowledge into their problems. However, prior knowledge is often limited in the black-box situation due to the black-box relation between $\boldsymbol{x}$ and $f(\boldsymbol{x})$. Hence, algorithm selection, as well as hyper-parameter tuning, is a tedious task for practitioners who are typically not experts in search algorithms.

Covariance Matrix Adaptation Evolution Strategy (CMA-ES), developed by Hansen and Ostermeier (2001); Hansen et al. (2003); Hansen and Kern (2004); Jastrebski and Arnold (2006), is recognized as a state-of-the-art derivative-free search algorithm for *difficult* continuous optimization problems (Rios and Sahinidis, 2013). CMA-ES is a stochastic and comparison-based search algorithm that maintains the multivariate normal distribution as a sampling distribution of candidate solutions. The distribution parameters such as the mean vector and the covariance matrix are updated at each iteration based on the candidate solutions and their objective value ranking, so that the sampling distribution will produce promising candidate solutions more likely in the next algorithmic iteration. The update of the distribution parameters is partially found as the natural gradient ascent on the manifold of the distribution parameter space equipped with the Fisher metric (Akimoto et al., 2010; Glasmachers et al., 2010; Ollivier et al., 2017), thereby revealing the connection to natural evolution strategies Wierstra et al. (2008); Sun et al. (2009); Glasmachers et al. (2010); Wierstra et al. (2014), whose parameter update is derived explicitly from the natural gradient principle.

Invariance (Hansen, 2000; Hansen et al., 2011) is one of the governing principles of the design of CMA-ES and the essence of its success on *difficult* continuous optimization problems consisting of ruggedness, ill-conditioning, and non-separability. CMA-ES exhibits several invariance properties such as invariance to order preserving transformation of the objective function, invariance to translation, rotation and coordinate-wise scaling of the search space (Hansen and Auger, 2014). Invariance guarantees the algorithm to work identically on an original problem and its transformed version. Thanks to its invariance properties, CMA-ES works, after an adaptation phase, equally well on separable and well-conditioned functions, which are easy for most search algorithms, and on non-separable and ill-conditioned functions produced by an affine coordinate transformation of the former, which are considered difficult for many other search algorithms. This also contributes to allow default hyper-parameter values to depend solely on the search space dimension and the population size, whereas many other search algorithms need tuning depending on problem difficulties to make the algorithms efficient (Karafotias et al., 2015).

On the other hand, exploiting problem structure, if possible, is beneficial for optimization speed. Different variants of CMA-ES have been proposed to exploit problem structure such as separability and limited variable dependency. They aim to achieve a better scaling with the dimension (Knight and Lunacek, 2007; Ros and Hansen, 2008; Akimoto et al., 2014; Akimoto and Hansen, 2016; Loshchilov, 2017). However, they lose

some of the invariance properties that CMA-ES has and compromise the performance on problems where their specific, more or less restrictive assumptions on the problem structure do not hold. For instance, the separable CMA-ES (Ros and Hansen, 2008) reduces the degrees of freedom of the covariance matrix from $(n^2+n)/2$ to n by adapting only the diagonal elements of the covariance matrix. It scales better in terms of internal time and space complexity and in terms of number of function evaluations to adapt the coordinate-wise variance of the search distribution. Good results are hence observed on functions with weak variable dependencies. However, unsurprisingly, the convergence speed of the separable CMA is significantly deteriorated on non-separable and ill-conditioned functions, where the shape of the level sets of the objective function can not be reasonably well approximated by the equi-probability ellipsoid defined by the normal distribution with diagonal (co)variance matrix.

In this paper, we aim to improve the performance of CMA-ES on a relatively wide class of problems by exploiting problem structure, however crucially, without compromising the performance on more difficult problems without this structure.[1]

The first mechanism we are concerned with is the so-called active covariance matrix update, which was originally proposed for the (μ, λ)-CMA-ES with intermediate recombination (Jastrebski and Arnold, 2006), and later incorporated into the $(1 + 1)$-CMA-ES (Arnold and Hansen, 2010). It utilizes unpromising solutions to actively decrease the eigenvalues of the covariance matrix. The active update consistently improves the adaptation speed of the covariance matrix in particular on functions where a low dimensional subspace dominates the function value. The positive definiteness of the covariance matrix is however not guaranteed when the active update is utilized. Practically, a small enough learning rate of the covariance matrix is sufficient to keep the covariance matrix positive definite with overwhelming probability, however, we would like to increase the learning rate when the population size is large. We propose two novel schemes that guarantee the positive definiteness of the covariance matrix, so that we take advantage of the active update even when a large population size is desired, e.g., when the objective function evaluations are distributed on many CPUs or when the objective function is rugged.

The main contribution of this paper is the diagonal acceleration of CMA by means of *adaptive diagonal decoding*, referred to as dd-CMA. We introduce a coordinate-wise variance matrix, $\mathbf{D}^2$, of the sampling distribution alongside the positive definite symmetric matrix $\mathbf{C}$, such that the resulting covariance matrix of the sampling distribution is represented by $\mathbf{DCD}$. We call $\mathbf{D}$ the diagonal decoding matrix. We update $\mathbf{C}$ with the original CMA, whereas $\mathbf{D}$ is updated similarly to separable CMA. An important point is that we want to update $\mathbf{D}$ faster than $\mathbf{C}$, by setting higher learning rates for the $\mathbf{D}$ update. However, when $\mathbf{C}$ contains non-zero covariances, the update of $\mathbf{D}$ can result in a drastic change of the sampling distribution and disturb the adaptation of $\mathbf{C}$. To resolve this issue, we introduce an adaptive damping mechanism for the $\mathbf{D}$ update, so that the difference (e.g., Kullback-Leibler divergence) between the distributions before and after the update remains sufficiently small. With this damping, $\mathbf{D}$ is updated as fast as by separable CMA on a separable function if the correlation matrix of the sampling distribution is close to the identity, and it suppresses the $\mathbf{D}$ update when the correlation matrix is away from the identity, i.e., variable dependencies have been learned.

The update of $\mathbf{D}$ breaks the rotation invariance of the original CMA, hence we

[1] Any covariance matrix, $\mathbf{\Sigma}$, can be uniquely decomposed into $\mathbf{\Sigma} = \mathbf{DCD}$, where $\mathbf{D}$ is a diagonal matrix and $\mathbf{C}$ is a correlation matrix. The addressed problem class can be characterized in that for the best problem approximation $\mathbf{\Sigma} = \mathbf{DCD}$ *both matrices,* $\mathbf{C}$ *and* $\mathbf{D}$*, have non-negligible condition number, say no less than* 100.

lose a mathematical guarantee of the original CMA that it performs identical on functions in an equivalence group defined by this invariance property. The ultimate aim of this work is to gain significant speed-up in some situations and to preserve the performance of the original CMA in the worst case. Functions where we expect that the diagonally accelerated CMA outperforms the original one have variables with different sensitivities, i.e., coordinate-wise ill-conditioning. Such functions may often appear in practice, since variables in a black-box continuous optimization problem can have distinct meanings. Diagonal acceleration however can even be superior to the optimal additive portfolio of the original CMA and the separable CMA. We demonstrate in numerical experiments that dd-CMA outperforms the original CMA not only on separable functions but also on non-separable ill-conditioned functions with coordinate-wise ill-conditioning that separable CMA can not efficiently solve.

The last contribution is a set of improved and simplified default parameter settings for the covariance matrix adaptation and for the cumulation factor for the so-called evolution path. These learning rates, whose default values have been previously expressed with independent formulas, are reformulated so that their dependencies are clearer. The new default learning rates also improve the adaptation speed of the covariance matrix on high dimensional problems without compromising stability.

The rest of this paper is organized as follows. We introduce the original and separable CMA-ES in Section 2. The active update of the covariance matrix with positive definiteness guarantee is proposed in Section 3. The adaptive diagonal decoding mechanism is introduced in Section 4. Section 5 is devoted to explain the renewed default hyper-parameter values for the covariance matrix adaptation and provides an algorithm summary of dd-CMA-ES and a link to publicly available Python code. Numerical experiments are conducted in Section 6 to see how effective each component of CMA with diagonal acceleration works in different situations. We conclude the paper in Section 7.

2 Evolution Strategy with Covariance Matrix Adaptation

We summon up the (μ_w, λ)-CMA-ES consisting of weighted recombination, cumulative step-size adaptation, and rank-one and rank-μ covariance matrix adaptation.

The CMA-ES maintains the multivariate normal distribution, $\mathcal{N}(\boldsymbol{m}, \sigma^2\mathbf{DCD})$, where $\boldsymbol{m} \in \mathbb{R}^n$ is the mean vector that represents the center of the search distribution, $\sigma \in \mathbb{R}_+$ is the so-called step-size that represents the scaling factor of the distribution spread, and $\mathbf{C} \in \mathbb{R}^{n\times n}$ is a positive definite symmetric matrix that represents the shape of the distribution ellipsoid. Though the covariance matrix of the sampling distribution is $\sigma^2\mathbf{DCD}$, we often call $\mathbf{C}$ the covariance matrix in the context of CMA-ES. In this paper, we apply this terminology, and we will call $\sigma^2\mathbf{DCD}$ the covariance matrix of the sampling distribution to distinguish them when necessary. The positive definite diagonal matrix $\mathbf{D} \in \mathbb{R}^{n\times n}$ is regarded as a diagonal decoding matrix, which represents the scaling factor of each design variable. It is fixed during the optimization, usually $\mathbf{D} = \mathbf{I}$, and does not appear in the standard terminology. However, it plays an important role in this paper, and we define the CMA-ES with $\mathbf{D}$ for the later modification.

2.1 Original CMA-ES

The CMA-ES repeats the following steps until it meets one or more termination criteria:

1. Sample λ candidate solutions, $\boldsymbol{x}_i$, independently from $\mathcal{N}(\boldsymbol{m}, \sigma^2\mathbf{DCD})$;
2. Evaluate candidate solutions on the objective, $f(\boldsymbol{x}_i)$, and sort them in ascending

order[2], $f(\boldsymbol{x}_{1:\lambda}) \leq \cdots \leq f(\boldsymbol{x}_{\lambda:\lambda})$, where $i:\lambda$ is the index of the i-th best candidate solution among $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_\lambda$;

3. Update the distribution parameters, $\boldsymbol{m}$, σ, and $\mathbf{C}$.

Sampling and Recombination To generate candidate solutions, we compute the (unique, symmetric) square root of the covariance matrix $\sqrt{\mathbf{C}}$ obeying $\mathbf{C}^{(t)} = \left(\sqrt{\mathbf{C}}\right)^2$. The candidate solutions are the affine transformation of independent and standard normally distributed random vectors,

$$\begin{aligned} \boldsymbol{z}_i &\sim \mathcal{N}(\mathbf{0}, \mathbf{I}) \ , \\ \boldsymbol{y}_i &= \sqrt{\mathbf{C}}\boldsymbol{z}_i \sim \mathcal{N}(\mathbf{0}, \mathbf{C}) \ , \\ \boldsymbol{x}_i &= \boldsymbol{m}^{(t)} + \sigma^{(t)}\mathbf{D}\boldsymbol{y}_i \sim \mathcal{N}(\boldsymbol{m}^{(t)}, (\sigma^{(t)})^2\mathbf{DCD}) \ . \end{aligned} \tag{1}$$

The default population size is $\lambda = 4 + \lfloor 3 \ln n \rfloor$. To reduce the time complexity per f-call without compromising the performance, we compute the matrix decomposition $\mathbf{C}^{(t)} = \left(\sqrt{\mathbf{C}}\right)^2$ every $t_{\text{eig}} = \max\left(1, \left\lfloor (\beta_{\text{eig}}(c_1 + c_\mu))^{-1} \right\rfloor\right)$ iteration, where c_1 and c_μ are the learning rates for the covariance matrix adaptation that appear later, and $\beta_{\text{eig}} = 10n$. If the learning rates are small enough ($c_1 + c_\mu \leqslant (2\beta_{\text{eig}})^{-1}$), the covariance matrix is regarded as insignificantly changing in each iteration and we stall the decomposition.

The mean vector $\boldsymbol{m}$ is updated by taking the weighted average of the promising candidate directions,

$$\boldsymbol{m}^{(t+1)} = \boldsymbol{m}^{(t)} + c_m \sum_{i=1}^{\mu} w_i (\boldsymbol{x}_{i:\lambda} - \boldsymbol{m}) \ , \tag{2}$$

where c_m is the learning rate for the mean vector update, usually $c_m = 1$. The number of promising candidate solutions are denoted by μ, and $\left(w_i\right)_{i=1}^{\mu}$ are recombination weights satisfying $w_i > 0$ for $i \leqslant \mu$. A standard choice is $w_i \propto \ln \frac{\lambda+1}{2} - \ln i$ for $i = 1, \ldots, \mu$ and $\mu = \lfloor \lambda/2 \rfloor$ and $\sum_{i=1}^{\mu} w_i = 1$.

Step-Size Adaptation The cumulative step-size adaptation (CSA) updates the step-size σ based on the length of the evolution path that accumulates the mean shift normalized by the current distribution covariance, i.e.,[3]

$$\boldsymbol{p}_\sigma^{(t+1)} = (1 - c_\sigma)\boldsymbol{p}_\sigma^{(t)} + \sqrt{c_\sigma(2 - c_\sigma)\mu_w} \sum_{i=1}^{\mu} w_i \boldsymbol{z}_{i:\lambda} \ , \tag{3}$$

$$\gamma_\sigma^{(t+1)} = (1 - c_\sigma)^2 \gamma_\sigma^{(t)} + c_\sigma(2 - c_\sigma) \ , \tag{4}$$

where $\mu_w = 1/\sum_{i=1}^{\mu} w_i^2$ is the so-called variance effective selection mass, and c_σ is the inverse time horizon for the $\boldsymbol{p}_\sigma$ update, aka cumulation factor. The scalar $\gamma_\sigma^{(t+1)}$ is a

[2] Ties, $f(\boldsymbol{x}_{i:\lambda}) = \cdots = f(\boldsymbol{x}_{i+k-1:\lambda})$, are treated by redistributing the averaged recombination weights $\sum_{l=0}^{k-1} w_{i+l}/k$ to tied solutions $\boldsymbol{x}_{i:\lambda}, \ldots, \boldsymbol{x}_{i+k-1:\lambda}$.

[3] When $\mathbf{D}$ is not the identity, (3) is not exactly equivalent to the original CSA (Hansen and Ostermeier, 2001): $\boldsymbol{z}_{i:\lambda} = (\mathbf{D}\sqrt{\mathbf{C}})^{-1}(\boldsymbol{x}_{i:\lambda} - \boldsymbol{m})/\sigma$ in this paper whereas originally $\boldsymbol{z}_{i:\lambda} = \sqrt{\mathbf{DCD}}^{-1}(\boldsymbol{x}_{i:\lambda} - \boldsymbol{m})/\sigma$. This difference results in rotating the second term of the RHS of (3) at each iteration with a different orthogonal matrix, and ends up in a different $\|\boldsymbol{p}_\sigma\|$. Krause et al. (2016) have theoretically studied the effect of this difference and argued that this systematic difference becomes small if the parameterization of the covariance matrix of the sampling distribution is unique and it converges up to scale. If $\mathbf{D}$ is fixed, the parameterization is unique. Later in this paper, we update both $\mathbf{D}$ and $\mathbf{C}$ but we force the parameterization to be unique by (34) and (35). Hence the systematic difference is expected to be small. See Krause et al. (2016) for details.

correction factor for small t and converges to 1, where $\gamma_\sigma^{(0)} = \|\boldsymbol{p}_\sigma^{(0)}\|^2/n = 0$.[4] The log step-size is updated as

$$\ln \sigma^{(t+1)} = \ln \sigma^{(t)} + \frac{c_\sigma}{d_\sigma} \left(\frac{\|\boldsymbol{p}_\sigma^{(t+1)}\|}{\chi_n} - \sqrt{\gamma_\sigma^{(t+1)}} \right) \ , \tag{5}$$

where d_σ is the damping factor for the σ update and $\chi_n = \sqrt{n}\big(1 - \frac{1}{4n} + \frac{1}{21n^2}\big)$ is an approximation of the expected value of the norm of an n-dimensional standard normally distributed random vector, $\sqrt{2}\Gamma\big(\frac{n+1}{2}\big)/\Gamma\big(\frac{n}{2}\big)$. The default values for c_σ and d_σ are

$$c_\sigma = \frac{\mu_w + 2}{n + \mu_w + 5} \tag{6}$$

$$d_\sigma = 1 + c_\sigma + 2 \max\left(0, \ \sqrt{\frac{\mu_w - 1}{n+1}} - 1\right) \ . \tag{7}$$

The damping parameter d_σ is introduced to stabilize the step-size adaptation when the population size is large (Hansen and Kern, 2004). When the step-size becomes too small by accident or is initialized so, the norm of the evolution path will become $O\big(\sqrt{\mu_w/n}\big)$, which results in a quick increase of σ if $d_\sigma = 1$ (Akimoto et al., 2008). For large μ_w, the chosen damping factor d_σ prevents an unreasonable increase of σ at the price of a reduced convergence speed. In case of $\mu_w \gg n$, the covariance matrix adaptation is the main component decreasing the overall variance of the sampling distribution, while the CSA is still effective to increase σ when necessary.

Covariance Matrix Adaptation The covariance matrix $\mathbf{C}$ is updated by the following formula that combines rank-one and rank-μ update

$$\begin{aligned} \mathbf{C}^{(t+1)} = &\left(1 - c_1\gamma_c^{(t+1)} - c_\mu \sum_{i=1}^{\mu} w_i\right) \mathbf{C}^{(t)} \\ &+ c_1 \left(\mathbf{D}^{-1}\boldsymbol{p}_c^{(t+1)}\right)\left(\mathbf{D}^{-1}\boldsymbol{p}_c^{(t+1)}\right)^{\mathsf{T}} + c_\mu \sum_{i=1}^{\mu} w_i \boldsymbol{y}_{i:\lambda}\boldsymbol{y}_{i:\lambda}^{\mathsf{T}} \ , \end{aligned} \tag{8}$$

where c_1 and c_μ are the learning rates for the rank-one update (2nd term) and the rank-μ update (3rd term), respectively, $\boldsymbol{p}_c$ is the evolution path that accumulates the successive mean movements and γ_c is the correction factor for the rank-one update, which are updated as

$$\boldsymbol{p}_c^{(t+1)} = (1 - c_c)\boldsymbol{p}_c^{(t)} + h_\sigma^{(t+1)} \sqrt{c_c(2 - c_c)\mu_w} \sum_{i=1}^{\mu} w_i \mathbf{D}\boldsymbol{y}_{i:\lambda} \ , \tag{9}$$

$$\gamma_c^{(t+1)} = (1 - c_c)^2\gamma_c^{(t)} + h_\sigma^{(t+1)} c_c(2 - c_c) \ , \tag{10}$$

where c_c is the inverse time horizon for the $\boldsymbol{p}_c$ update. The Heaviside function $h_\sigma^{(t+1)}$ is introduced to stall the update of $\boldsymbol{p}_c$ if $\|\boldsymbol{p}_\sigma\|$ is large, i.e., when the step-size is rapidly increasing. It is defined as

$$h_\sigma^{(t+1)} = \begin{cases} 1 & \text{if } \|\boldsymbol{p}_\sigma^{(t+1)}\|^2/\gamma_\sigma^{(t+1)} < \big(2 + \frac{4}{n+1}\big)n \\ 0 & \text{otherwise} \end{cases} . \tag{11}$$

[4] An elegant alternative to introducing γ_σ is to use $c_\sigma^{(t)} = \max(c_\sigma, 1/t)$ in place of c_σ in (3), assuming the first $t = 1$. This resembles a simple average while $t \leq 1/c_\sigma$ and only afterwards discounts older information by the original decay of $1 - c_\sigma$.

The default parameters for c_1, c_μ, and c_c are smaller than one and presented later.

2.2 Separable Covariance Matrix Adaptation

The separable covariance matrix adaptation (sep-CMA, Ros and Hansen (2008)) adapts only the coordinate-wise variance of the sampling distribution, i.e., the diagonal elements of the covariance matrix in the same way as (8), but with larger learning rates. In our notation scheme, we keep $\mathbf{C}$ to be the identity and describe sep-CMA by updating $\mathbf{D}$. The update of coordinate k follows

$$[\mathbf{D}^{(t+1)}]_{k,k}^2 = [\mathbf{D}^{(t)}]_{k,k}^2 \Bigg(1 - c_1 \gamma_c^{(t+1)} - c_\mu \sum_{i=1}^{\mu} w_i + c_1 [\boldsymbol{p}_c^{(t+1)}]_k^2 / [\mathbf{D}^{(t)}]_{k,k}^2 + c_\mu \sum_{i=1}^{\mu} w_i \, [\boldsymbol{z}_{i:\lambda}]_k^2 \Bigg) \quad . \tag{12}$$

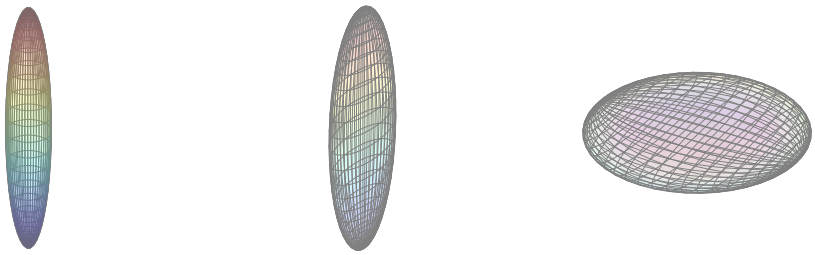
The learning rates c_1 and c_μ are set differently from those used for the $\mathbf{C}$ update.

One advantage of the separable CMA is that all operations can be done in linear time per f-call. Therefore, it is promising if f-calls are cheap. The other advantage is that one can set the learning rate greater than those used for the $\mathbf{C}$ update, since the degrees of freedom of the covariance matrix of the sampling distribution is n, rather than $n(n+1)/2$. The adaptation speed of the covariance matrix is faster than for the original CMA. However, if the problem is non-separable and has strong variable dependencies, adapting the coordinate-wise scaling is not enough to make the search efficient. More concisely, if the inverse Hessian of the objective function, $\mathrm{Hess}(f)^{-1}$, is not well approximated by a diagonal matrix, the convergence speed will be $O(1/\operatorname{Cond}(\mathbf{D}^2\mathrm{Hess}(f)))$, which is empirically observed on convex quadratic functions as well as theoretically deduced in Akimoto et al. (2018). In practice, it is rarely known in advance whether the separable CMA is appropriate or not. Ros and Hansen (2008) propose to use the separable CMA for hundred times dimension function evaluations and then switch to the original CMA afterwards. Such an algorithm has been benchmarked in Ros (2009), where the first $1 + 100n/\sqrt{\lambda}$ iterations are used for the separable CMA.

3 Active covariance matrix update with guarantee of positive definiteness

Active covariance matrix adaptation (referred to as Active-CMA, Jastrebski and Arnold, 2006) utilizes the information of unpromising solutions, $\boldsymbol{x}_{i:\lambda}$ for $i > \mu$, to update the covariance matrix. The modification is rather simple. We prepare λ recombination weights $(w_i)_{i=1}^{\lambda}$ for the active covariance matrix update, where w_i are not anymore restricted to be positive. For example, the recombination weights used in Jastrebski and Arnold (2006) are

$$w_i = \begin{cases} 1/\mu & (i \leqslant \mu) \\ 0 & (\mu < i \leqslant \lambda - \mu) \\ -1/\mu & (\lambda - \mu < i) \end{cases} \tag{13}$$



(a) Cigar type (b) Ellipsoid type (c) Discus type

Figure 1: Levelset of three convex quadratic functions $\boldsymbol{x}^\mathsf{T}\mathbf{H}\boldsymbol{x}$ with Hessian $\mathbf{H}$ of condition number 30. Cigar type: $\mathbf{H} = \mathrm{diag}(30, 30, 1)$, Ellipsoid type: $\mathbf{H} = \mathrm{diag}(1, \sqrt{30}, 30)$, Discus type: $\mathbf{H} = \mathrm{diag}(1, 1, 30)$.

The update formula (8) is then replaced with

$$\mathbf{C}^{(t+1)} = \left(1 - c_1\gamma_c^{(t+1)} - c_\mu \sum_{i=1}^{\lambda} w_i\right)\mathbf{C}^{(t)} + c_1\left(\mathbf{D}^{-1}\boldsymbol{p}_c^{(t+1)}\right)\left(\mathbf{D}^{-1}\boldsymbol{p}_c^{(t+1)}\right)^\mathsf{T} + c_\mu \sum_{i=1}^{\lambda} w_i \boldsymbol{y}_{i:\lambda}\boldsymbol{y}_{i:\lambda}^\mathsf{T} \ , \quad (14)$$

where shaded areas depict the difference to (8). The only difference is that we use all λ candidate solutions to update the covariance matrix with positive and negative recombination weights.[5]

Each component of the covariance matrix adaptation, rank-one update, rank-μ update, and active update, produces complementary effects. The rank-one update of the covariance matrix (the second term on the RHS of (14), Hansen and Ostermeier, 2001) accumulates the successive steps of the distribution mean and increases the eigenvalues of the covariance matrix in the direction of the successive movements. It excels at learning one long axis of the covariance matrix, and is effective on functions with a subspace of a relatively small dimension where the function value is less sensitive than in its orthogonal subspace. Figure 1a visualizes an example of such function. See also Figure 7 in Hansen and Auger (2014) for numerical results. On the other hand, since the update is always of rank one, the learning rate c_1 needs to be sufficiently small to keep the covariance matrix regular and stable.

The rank-μ update (the third term on the RHS of (14) with positive w_i, Hansen et al., 2003) utilizes the information of μ successful candidate solutions in a different way than the rank-one update. It computes the empirical covariance matrix of successful mutation steps $\boldsymbol{y}_i$. The update matrix is with probability one of rank $\min(n, \mu)$, allowing to set a relatively large learning rate c_μ. It reduces the number of iterations to adapt the covariance matrix when $\lambda \gg 1$.

[5] As of 2018, many implementations of CMA-ES feature the active update of $\mathbf{C}$ and it should be considered as default.

Both, rank-one and rank-μ update, try to increase the variances in the subspace of successful mutation steps. The eigenvalues corresponding to unsuccessful mutation steps only passively fade out. The active update actively decreases such eigenvalues. It consistently accelerates covariance matrix adaptation, and the improvement is particularly pronounced on functions with a small number of dominating eigenvalues of the Hessian of the objective function. Figure 1c is an example of such function.

A disadvantage of the active update with negative recombination weights such as (13) is to have no guarantee of the positive definiteness of the covariance matrix anymore. It is easy to see that the rank-one and rank-μ update of CMA in (8) guarantee that the minimal eigenvalue of $\mathbf{C}^{(t+1)}$ is no smaller than the minimal eigenvalue of $\mathbf{C}^{(t)}$ times $1-c_1-c_\mu \sum_{i=1}^{\mu} w_i$, since the second and third terms only increase the eigenvalues. However, the introduction of negative recombination weights can violate the positive definiteness because the third term may decrease the minimum eigenvalue arbitrarily. Jastrebski and Arnold (2006) set a sufficiently small learning rate for the active update, i.e., the absolute values of the negative recombination weights sum up to a smaller value than one. It will prevent the covariance matrix from being non-positive with high probability, but it does not *guarantee* positive definiteness. Moreover, it becomes ineffective when the population size is relatively large and a greater learning rate is desired.

Krause and Glasmachers (2015) apply the active update with positive definiteness guarantee by introducing the exponential covariance matrix update, called xCMA. Instead of updating the covariance matrix in an additive way as in (14), the covariance matrix is updated as

$$\mathbf{C}^{(t+1)} = \sqrt{\mathbf{C}^{(t)}} \exp\left(\Delta\right) \sqrt{\mathbf{C}^{(t)}} \ . \tag{15}$$

where Δ is a symmetric matrix. Since the eigenvalues of the matrix exponential are e^{δ_i} where δ_i are the eigenvalues of Δ, the positive definiteness is naturally guaranteed. Arnold and Hansen (2010) achieve the positive definiteness guarantee in the $(1+1)$-CMA-ES by rescaling the negative recombination weights depending on the norm of unsuccessful mutation steps $\|\boldsymbol{z}\|$. In this paper we introduce two strategies that are both considered as generalization of this idea to the (μ_w, λ)-CMA-ES.[6]

3.1 Method 1: Scaling Down the Update Factor

To guarantee the positive definiteness of the covariance matrix, we rescale the update factor of the covariance matrix so that the changes of the minimum eigenvalue of the covariance matrix is bounded. We start by introducing the rescaling of unpromising solutions,

$$\tilde{\boldsymbol{y}}_{i:\lambda} = \begin{cases} \boldsymbol{y}_{i:\lambda} & w_i \geqslant 0 \\ \frac{\sqrt{n}}{\|\boldsymbol{z}_{i:\lambda}\|} \boldsymbol{y}_{i:\lambda} & w_i < 0, \end{cases} \quad \text{and analogously,} \quad \tilde{\boldsymbol{z}}_{i:\lambda} = \begin{cases} \boldsymbol{z}_{i:\lambda} & w_i \geqslant 0 \\ \frac{\sqrt{n}}{\|\boldsymbol{z}_{i:\lambda}\|} \boldsymbol{z}_{i:\lambda} & w_i < 0. \end{cases} \tag{16}$$

[6] There are variants of CMA-ES that update a factored matrix $\mathbf{A}$ satisfying $\mathbf{C} = \mathbf{A}\mathbf{A}^\mathsf{T}$ (e.g., eNES by Sun et al., 2009). No matter how $\mathbf{A}$ is updated, the positive semi-definiteness of $\mathbf{C}$ is guaranteed since $\mathbf{A}\mathbf{A}^\mathsf{T}$ is always positive semi-definite. However this approach has a potential drawback that a negative update may end up increasing a variance. To see this, consider the case $\mathbf{A} \leftarrow \mathbf{A}(\mathbf{I} - \eta \boldsymbol{e}\boldsymbol{e}^\mathsf{T})$, where $\boldsymbol{e}$ is some vector and $\eta > 0$ represents the learning rate times the recombination weight. Then, the covariance matrix follows $\mathbf{C} \leftarrow \mathbf{A}(\mathbf{I} - \eta \boldsymbol{e}\boldsymbol{e}^\mathsf{T})^2\mathbf{A}^\mathsf{T}$. This update shrinks a variance if η is sufficiently small ($\eta < 1/\|\boldsymbol{e}\|^2$), however, it increases the variance if η is large ($\eta > 1/\|\boldsymbol{e}\|^2$). Hence, a negative update with a long vector $\boldsymbol{e}$ and/or a large η will cause an opposite effect. Therefore, the factored matrix update is not a conclusive solution to the positive definiteness issue.

This rescaling results in projecting unpromising solutions onto the surface of the hyperellipsoid defined by $\boldsymbol{y}^\mathsf{T}\mathbf{C}^{-1}\boldsymbol{y} = n$. By this projection we achieve three desired effects. First, the update becomes bounded which makes it easier to control positive definiteness. Second, short steps are elongated, enhancing their effect in the update. Third, long steps are shortened, reducing their effect in the update. The two latter effects counter the correlation between rank and length of steps in unfavorable directions. For any given unfavorable direction, longer steps in this direction are most likely ranked worse than shorter steps.

With these scaled solutions, the covariance matrix is updated as

$$\mathbf{C}^{(t+1)} = \mathbf{C}^{(t)} + \alpha^{(t)} \left(c_1 \left[\left(\mathbf{D}^{-1}\boldsymbol{p}_c^{(t+1)} \right) \left(\mathbf{D}^{-1}\boldsymbol{p}_c^{(t+1)} \right)^\mathsf{T} - \gamma_c^{(t+1)}\mathbf{C}^{(t)} \right] + c_\mu \sum_{i=1}^{\lambda} w_i \left[\tilde{\boldsymbol{y}}_{i:\lambda}\tilde{\boldsymbol{y}}_{i:\lambda}^\mathsf{T} - \mathbf{C}^{(t)} \right] \right) , \tag{17}$$

where shaded areas highlight the difference to (14) and $\alpha^{(t)} \leqslant 1$ is the scaling factor to guarantee the positive definiteness of the covariance matrix. Note that if $\alpha^{(t)} = 1$, it is equivalent to (14) except for the rescaling of the unpromising samples. Importantly, the rescaling of unpromising solutions does not affect the stationarity of the parameter update, i.e., $\mathbb{E}[\mathbf{C}^{(t+1)} \mid \mathbf{C}^{(t)}] = \mathbf{C}^{(t)}$ under a random function such as $f(\boldsymbol{x}) \sim \mathcal{U}(0,1)$. This is shown as follows. First, given $\boldsymbol{p}_c^{(t)} \sim \mathcal{N}(\mathbf{0}, \gamma_c^{(t)}\mathbf{D}\mathbf{C}^{(t)}\mathbf{D})$, it is easy to see that $\boldsymbol{p}_c^{(t+1)} \sim \mathcal{N}(\mathbf{0}, \gamma_c^{(t+1)}\mathbf{D}\mathbf{C}^{(t)}\mathbf{D})$. Second, using $\mathbb{E}[\boldsymbol{z}\boldsymbol{z}^\mathsf{T}/\|\boldsymbol{z}\|^2] = \mathbf{I}/n$ (Lemma 1 of Heijmans, 1999) we have $\mathbb{E}\big[\sum_{i=1}^{\lambda} w_i \tilde{\boldsymbol{y}}_{i:\lambda}\tilde{\boldsymbol{y}}_{i:\lambda}^\mathsf{T} \mid \mathbf{C}^{(t)}\big] = \big(\sum_{i=1}^{\lambda} w_i\big)\mathbf{C}^{(t)}$. Finally combining them, we obtain $\mathbb{E}[\mathbf{C}^{(t+1)} \mid \mathbf{C}^{(t)}] = \mathbf{C}^{(t)}$.

The main idea is to scale down, if necessary, the update of the covariance matrix by setting $\alpha^{(t)} < 1$ in (17). To provide a better intuition, we start by considering the case $t_{\mathrm{eig}} = 1$, i.e., $\mathbf{C}^{(t)} = \sqrt{\mathbf{C}}^2$ for every iteration t. Equation (17) can be written as

$$\mathbf{C}^{(t+1)} = \sqrt{\mathbf{C}} \left[\mathbf{I} + \alpha^{(t)}\mathbf{Z}^{(t)} \right] \sqrt{\mathbf{C}} \tag{18}$$

with

$$\mathbf{Z}^{(t)} = c_1 \left[\left(\sqrt{\mathbf{C}}^{-1}\mathbf{D}^{-1}\boldsymbol{p}_c^{(t+1)} \right) \left(\sqrt{\mathbf{C}}^{-1}\mathbf{D}^{-1}\boldsymbol{p}_c^{(t+1)} \right)^\mathsf{T} - \gamma_c^{(t+1)}\mathbf{I} \right] + c_\mu \sum_{i=1}^{\lambda} w_i \left[\tilde{\boldsymbol{z}}_{i:\lambda}\tilde{\boldsymbol{z}}_{i:\lambda}^\mathsf{T} - \mathbf{I} \right] . \tag{19}$$

Then, the scaling down parameter $\alpha^{(t)}$ in (18) is taken so that $\mathbf{I} + \alpha^{(t)}\mathbf{Z}^{(t)}$ is positive definite. Then, with maximum and minimum eigenvalue of a matrix $\mathbf{A}$ denoted by $d_1(\mathbf{A})$ and $d_n(\mathbf{A})$, respectively, we have that

$$d_1(\mathbf{I} + \alpha^{(t)}\mathbf{Z}^{(t)})\mathbf{C} \succcurlyeq \sqrt{\mathbf{C}}\big[\mathbf{I} + \alpha^{(t)}\mathbf{Z}^{(t)}\big]\sqrt{\mathbf{C}} \succcurlyeq d_n(\mathbf{I} + \alpha^{(t)}\mathbf{Z}^{(t)})\mathbf{C} , \tag{20}$$

where $\mathbf{A} \succcurlyeq \mathbf{B}$ mean that $\mathbf{A} - \mathbf{B}$ is positive semi-definite (i.e., all eigenvalues are non-negative)[7] for any two square matrices $\mathbf{A}$ and $\mathbf{B}$. Moreover, $d_n(\mathbf{I} + \alpha^{(t)}\mathbf{Z}^{(t)}) =$

[7] Some references (e.g., Harville (2008)) use the term "positive semi-definite matrix" for a matrix with positive and zero eigenvalues and "non-negative definite matrix" is used for matrices with non-negative eigenvalues, whereas in other references these terms are used for matrices with non-negative eigenvalues. In this paper, we apply the latter terminology.

$1 + \alpha^{(t)} d_n(\mathbf{Z}^{(t)})$. Therefore, if $\alpha^{(t)} < 1/|d_n(\mathbf{Z}^{(t)})|$, the resulting covariance matrix is guaranteed to be positive definite.

For the general case ($t_{\mathrm{eig}} \geqslant 1$), let t be the iteration at which the last eigen decomposition was performed, i.e., $\mathbf{C}^{(t)} = \sqrt{\mathbf{C}}^2$. For iterations $k \in [t, t + t_{\mathrm{eig}})$, we store the intermediate update matrix $\mathbf{Z}^{(k)}$. The covariance matrix is updated only when the eigen decomposition is required, i.e., at iteration $t + t_{\mathrm{eig}}$, as

$$\mathbf{C}^{(t+t_{\mathrm{eig}})} = \sqrt{\mathbf{C}} \left[\mathbf{I} + \alpha^{(t+t_{\mathrm{eig}})} \sum_{k=t}^{t+t_{\mathrm{eig}}-1} \mathbf{Z}^{(k)} \right] \sqrt{\mathbf{C}} \ . \tag{21}$$

Analogously to the above argument, the resulting covariance matrix is positive definite if the inside of the brackets is positive definite. We set the scaling down parameter as

$$\alpha^{(t+t_{\mathrm{eig}})} = \min \left(\frac{0.75}{\left| d_n\big(\sum_{k=t}^{t+t_{\mathrm{eig}}-1} \mathbf{Z}^{(k)} \big) \right|}, \ 1 \right) \ . \tag{22}$$

The first argument guarantees that the minimum eigenvalue of $\mathbf{C}^{(t+t_{\mathrm{eig}})}$ is greater than or equal to $1/4$th of the minimum eigenvalue of $\mathbf{C}^{(t)}$. More concisely, we have $\mathbf{C}^{(t+t_{\mathrm{eig}})} \succcurlyeq \frac{1}{4}\mathbf{C}^{(t)}$. The last argument, which is the most frequent case in practice, implies that the covariance matrix update does not need to be scaled down.

3.2 Method 2: Scaling Down Negative Weights

An alternative way to guarantee the positive definiteness of the covariance matrix is to scale down the sum of the absolute values of the negative weights, combined with the projection of unpromising solutions introduced above. We use the same update formula (21), but $\alpha^{(t+t_{\mathrm{eig}})} = 1$.

The positive definiteness of the covariance matrix is guaranteed under the condition $1/t_{\mathrm{eig}} > c_1 + c_\mu \sum_{i=1}^{\lambda} w_i + n c_\mu \sum_{i:\ w_i<0} |w_i|$. More precisely, we have the following claim.

Theorem 3.1. *If* $1/t_{\mathrm{eig}} > c_1 + c_\mu \sum_{i=1}^{\lambda} w_i + n c_\mu \sum_{i:\ w_i<0} |w_i|$, *then* $\mathbf{C}^{(t+t_{\mathrm{eig}})} \succcurlyeq \big(1 - t_{\mathrm{eig}}\big(c_1 + c_\mu \sum_{i=1}^{\lambda} w_i + n c_\mu \sum_{i:\ w_i<0} |w_i|\big)\big)\mathbf{C}^{(t)}$.

Proof of Theorem 3.1. First, we consider the case $t_{\mathrm{eig}} = 1$. Using the fact that the self outer product $\mathbf{v}\mathbf{v}^{\mathsf{T}}$ of a vector $\mathbf{v}$ is a matrix of rank 1 with the only nonzero eigenvalue of $\|\mathbf{v}\|^2$, we have

$$\begin{aligned}
\mathbf{C}^{(t+1)} &\succcurlyeq \left(1 - c_1 - c_\mu \sum_{i=1}^{\lambda} w_i\right) \mathbf{C}^{(t)} - c_\mu \sum_{i:\ w_i<0} |w_i| \frac{n \boldsymbol{y}_{i:\lambda} \boldsymbol{y}_{i:\lambda}^{\mathsf{T}}}{\|\boldsymbol{z}_{i:\lambda}\|^2} \\
&= \sqrt{\mathbf{C}} \left[\left(1 - c_1 - c_\mu \sum_{i=1}^{\lambda} w_i\right) \mathbf{I} - c_\mu \sum_{i:\ w_i<0} n |w_i| \frac{\boldsymbol{z}_{i:\lambda} \boldsymbol{z}_{i:\lambda}^{\mathsf{T}}}{\|\boldsymbol{z}_{i:\lambda}\|^2} \right] \sqrt{\mathbf{C}} \\
&\succcurlyeq \sqrt{\mathbf{C}} \left[\left(1 - c_1 - c_\mu \sum_{i=1}^{\lambda} w_i - n c_\mu \sum_{i:\ w_i<0} |w_i| \right) \mathbf{I} \right] \sqrt{\mathbf{C}} \\
&= \left(1 - c_1 - c_\mu \sum_{i=1}^{\lambda} w_i - n c_\mu \sum_{i:\ w_i<0} |w_i| \right) \mathbf{C}^{(t)} \ .
\end{aligned}$$

The analogous argument holds for $t_{\mathrm{eig}} > 1$ as well. This completes the proof. □

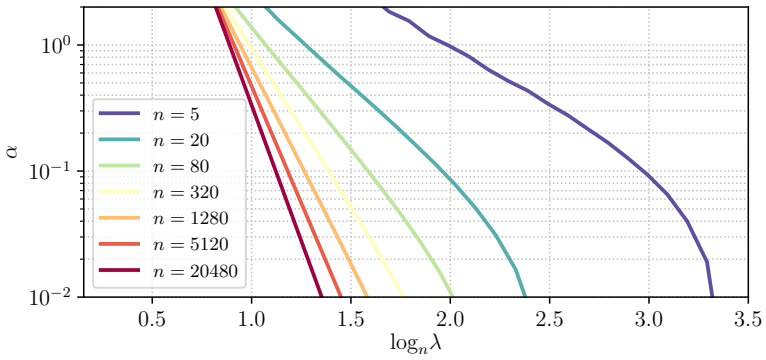


Figure 2: The correction factor α in (23).

We show how to construct the recombination weights so that they satisfy the sufficient condition of Theorem 3.1 as follows. Let $(w'_i)_{i=1}^{\lambda}$ be the pre-defined weights that are non-increasing. Without loss of generality, we assume that the first μ weights are positive and sum to one. The recombination weights are $w_i = w'_i$ for $w'_i \geqslant 0$ and $w_i = \alpha w'_i$ for $w'_i < 0$, where $\alpha \in (0, 1]$. Then, the sufficient condition in Theorem 3.1 reads $1/t_{\mathrm{eig}} > c_1 + c_\mu + (n-1)c_\mu \alpha \sum_{i:\, w_i<0} |w'_i|$. It holds if we set for example $t_{\mathrm{eig}} < (c_1 + c_\mu)^{-1}$, as satisfied by the default choice $t_{\mathrm{eig}} = \max\left(1, \left\lfloor (\beta_{\mathrm{eig}}(c_1 + c_\mu))^{-1} \right\rfloor\right)$, and

$$\alpha = \frac{1/t_{\mathrm{eig}} - (c_1 + c_\mu)}{n c_\mu \sum_{i:\, w_i<0} |w'_i|} \quad . \tag{23}$$

Then, it is guaranteed that $\mathbf{C}^{(t+t_{\mathrm{eig}})} \succcurlyeq \frac{1}{n}\mathbf{C}^{(t)}$.

This method is simpler than the method described in Section 3.1 since it does not require an additional eigenvalue computation. However, to guarantee the positive definiteness in this way, we bound the unrealistic worst case where all unpromising candidate solutions are sampled on a line. Therefore, the scaling down factor is set much smaller than the value that is actually needed in practice. Figure 2 visualizes the correction factor α under our choice of the default pre-weights and learning rates described in Section 5. The sum of the negative recombination weights needs to decrease as the population size increases. For $n \geqslant 320$, the factor is less than 0.1 for $\lambda \geqslant n^{1.5}$. Unless the internal computational time becomes a critical bottleneck, we prefer the method described in Section 3.1 over the method in Section 3.2.

3.3 Choice of Recombination Weights

We first review the rationale for the choice of the positive recombination weights. The default positive recombination weights, $\ln \frac{\lambda+1}{2} - \ln i$ for $i \leqslant \lambda/2$, are based on the quality gain analysis of the weighted recombination ES with isotropic Gaussian distribution (i.e., $\mathbf{C} \propto \mathbf{I}$). Arnold (2006) studied the quality gain on a spherical function, and derived the optimal recombination weights for the infinite dimensional case. They are

$$w_i \propto -\mathbb{E}[\mathcal{N}_{i:\lambda}] \quad \text{for all } i = 1, \ldots, \lambda, \tag{24}$$

where $\mathcal{N}_{1:\lambda} \leqslant \cdots \leqslant \mathcal{N}_{\lambda:\lambda}$ is the ordered statistics from the independent and standard normally distributed random variables $\mathcal{N}_1, \ldots, \mathcal{N}_\lambda$. Recently, it has been shown that the

same recombination weights are optimal for a general convex quadratic function with Hessian matrix $\mathbf{A}$ satisfying $\mathrm{Tr}(\mathbf{A}^2)/\mathrm{Tr}(\mathbf{A})^2 \ll 1$ for n large (Akimoto et al., 2018). The value $\ln \frac{\lambda+1}{2} - \ln i$ approximates the first half of the optimal recombination weights.[8]

We propose to use the recombination weights constructed as follows. Let

$$w'_i = \ln \frac{\lambda+1}{2} - \ln i \qquad \text{for } i = 1, \dots, \lambda. \tag{25}$$

The variance effective selection mass for the positive and negative weights is computed as

$$\mu_w = \frac{(\sum_{i:w'_i>0} |w'_i|)^2}{\sum_{i:w'_i>0} |w'_i|^2} \qquad \text{and} \qquad \mu_w^- = \frac{(\sum_{i:w'_i<0} |w'_i|)^2}{\sum_{i:w'_i<0} |w'_i|^2} \ . \tag{26}$$

The learning rates c_1 and c_μ may be computed depending on the above quantities. The default values for the learning rates are discussed in Section 5. The recombination weights w_i are set as follows

$$w_i = \begin{cases} \frac{w'_i}{\sum_{j:w'_j>0} |w'_j|} & \text{for } i \text{ with } w'_i \geqslant 0, \\ \frac{w'_i}{\sum_{j:w'_j<0} |w'_j|} \times \min\left(1 + \frac{c_1}{c_\mu},\ 1 + \frac{2\mu_w^-}{\mu_w+2}\right) & \text{for } i \text{ with } w'_i < 0. \end{cases} \tag{27}$$

The positive recombination weights are unchanged from the default settings without active CMA. They are approximately proportional to the positive half of the optimal recombination weights. However, this is not the case for the negative recombination weights. The optimal recombination weights are symmetric about zero, i.e. $w_i = -w_{\lambda-i+1}$, whereas our negative weights tend to level out for the following reasons. The above mentioned quality gain results consider only the mean update. The obtained optimal values are not necessarily optimal for the covariance matrix adaptation. Since we use only positive recombination weights for the mean vector update, the negative weights do not need to correspond to these optimal recombination weights. Furthermore, the optimal negative weights—greater absolute values for worse steps—counteract our motivation for rescaling of unpromising steps discussed in Section 3.1. Our choice of the negative recombination weights is a natural extension of the default positive recombination weights. The shape of our recombination weights is somewhat similar to the one in xCMA-ES (Krause and Glasmachers, 2015), where the first half is $w_i - 1/\lambda$ and the last half is $-1/\lambda$. A minor difference is that xCMA-ES assigns negative values even for some of the better half of the candidate solutions, whereas our setting assigns positive values only for the better half and negative values only for the worse half.

Positive and negative recombination weights are scaled differently. Positive weights are scaled to sum to one, whereas negative weights are scaled so that the sum of their absolute values is the minimum of $1 + \frac{c_1}{c_\mu}$ and $1 + \frac{2\mu_w^-}{\mu_w+2}$. The latter corrects for unbalanced positive and negative weights, but is usually greater than the former

[8] Interesting results are derived from the quality gain (and progress rate) analysis, see e.g. Section 4.2 of Hansen et al. (2015). Comparing the $(1+1)$-ES and (μ, λ)-ES with intermediate recombination, we realize that they reach the same quality gain in the limit for n to infinity when μ is the optimal value, $\mu \approx 0.27\lambda$ (Beyer, 1995). That is, a non-elitist strategy with optimal μ/λ for intermediate recombination can reach the same speed as the elitist strategy. With the optimal recombination weights above, the weighted recombination ES is 2.475 times faster than those algorithms. If we employ only the positive half of the optimal recombination weights, the speed will be halved, yet it is faster by the factor of 1.237.

with our default values. The former makes the decay factor $1 - c_1 - c_\mu \sum_{i=1}^{\lambda} w_i$ of the covariance matrix update (see (14)) to 1. That is, the covariance matrix does not passively shrink, and only the negative update can shrink the covariance matrix. Krause and Glasmachers (2015) mention to have no passive shrinkage by setting the sum of the weights to zero, but the effect of the rank-one update was not taken into account and thus in xCMA the passive decay factor of $1 - c_1$ remains.

4 Adaptive Diagonal Decoding (dd-CMA)

The separable CMA-ES (Ros and Hansen, 2008) enables to adapt a diagonal covariance matrix faster than the standard CMA-ES because the learning rates are $O(n)$ times greater than in standard CMA-ES. This works well on separable objective functions, where separable CMA-ES adapts the (co)variance matrix by a factor of $O(n)$ faster than CMA-ES. To accelerate standard CMA, we combine separable CMA and standard CMA. We adapt both $\mathbf{D}$ and $\mathbf{C}$ at the same time[9], where $\mathbf{D}$ is updated with *adaptive* learning rates which can be much greater than those used to update $\mathbf{C}$.

4.1 Algorithm

The update of $\mathbf{D}$ is similar to separable CMA, but is done in local exponential coordinates. The update of $\mathbf{D}$ is as follows

$$[\Delta_D]_{k,k} = c_{1,D}\big(\big[\sqrt{\mathbf{C}}^{-1}(\mathbf{D}^{(t)})^{-1}\boldsymbol{p}_{c,D}\big]_k^2 - \gamma_{c,D}\big) + c_{\mu,D}\sum_{i=0}^{\lambda} w_{i,D}\big(\big[\tilde{\boldsymbol{z}}_{i:\lambda}\big]_k^2 - 1\big) \tag{28}$$

$$\mathbf{D}^{(t+1)} \leftarrow \mathbf{D}^{(t)} \cdot \exp\left(\frac{\Delta_D}{2\beta^{(t)}}\right) \ , \tag{29}$$

where $[\Delta_D]_{k,k}$ is the k-th diagonal element of the diagonal matrix Δ_D, the evolution path $\boldsymbol{p}_{c,D}$ feeds the rank-one $\mathbf{D}$ update, $c_{1,D}$ and $c_{\mu,D}$ are the learning rates for the rank-one and rank-μ $\mathbf{D}$ updates, respectively, the recombination weights $w_{i,D}$ are computed by (27) using $c_{1,D}/c_{\mu,D}$ instead of c_1/c_μ, $\tilde{\boldsymbol{z}}_{i:\lambda}$ is defined in (16), and $\beta^{(t)}$ is the damping factor for the diagonal matrix update given in (31) below. The evolution path $\boldsymbol{p}_{c,D}$ and its normalization factor $\gamma_{c,D}$ are updated in the same way as (9) and (10) with the cumulation factor $c_{c,D}$ rather than c_c. The matrix exponential of a diagonal matrix is $\exp(\mathrm{diag}(d_1, \ldots, d_n)) = \mathrm{diag}(\exp(d_1), \ldots, \exp(d_n))$.

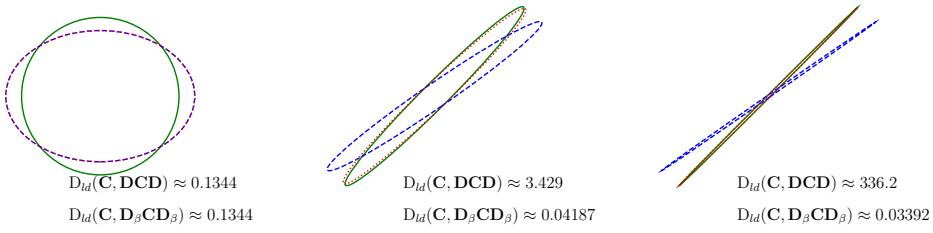
Using the correlation matrix of the covariance matrix $\mathbf{C}$ of the last time the matrix was decomposed,

$$\mathrm{corr}(\mathbf{C}) := \sqrt{\mathrm{diag}(\mathbf{C})}^{-1}\mathbf{C}\sqrt{\mathrm{diag}(\mathbf{C})}^{-1} \ , \tag{30}$$

where $\mathrm{diag}(\mathbf{C})$ is the diagonal matrix whose diagonal elements are the diagonal elements of $\mathbf{C}$, the damping factor is based on the square root of the condition number of this correlation matrix as

$$\beta^{(t)} = \max\left(1,\ \sqrt{\mathrm{Cond}\left(\mathrm{corr}(\mathbf{C})\right)} - \beta_{\mathrm{thresh}} + 1\right) \ , \tag{31}$$

[9]Considering the eigen decomposition of the covariance matrix, $\mathbf{E}\boldsymbol{\Lambda}\mathbf{E}^\mathsf{T}$, instead of the decomposition $\mathbf{DCD}$, our attempts to additionally adapt $\boldsymbol{\Lambda}$ were unsuccessful. We attribute the failure to the strong interdependency between $\boldsymbol{\Lambda}$ and $\mathbf{E}$. Compared to the eigen decomposition, the diagonal decoding model $\mathbf{DCD}$ also has the advantage to be interpretable as a rescaling of variables. We never considered the decomposition $\sqrt{\mathbf{C}}\mathbf{D}^2\sqrt{\mathbf{C}}$ as proposed by one of the reviewers of this paper, however, at first sight, we do not see any particular advantage in favor of this decomposition.



(a) axis ratio: 1 (b) axis ratio: 10 (c) axis ratio: 100

Figure 3: Equiprobability ellipse defined by $\mathbf{C}$, $\mathbf{DCD}$, and $\mathbf{D}_\beta\mathbf{CD}_\beta$. Green ellipses represent $\mathcal{N}(\mathbf{0}, \mathbf{C})$ with axis ratio of 1, 10, 100 and inclination angle $\pi/4$. Blue dashed ellipses represent $\mathcal{N}(\mathbf{0}, \mathbf{DCD})$ with $\mathbf{D} = [1.2, 1/1.2]$. Red ellipses represent $\mathcal{N}(\mathbf{0}, \mathbf{D}_\beta\mathbf{CD}_\beta)$ with $\mathbf{D}_\beta$ damped by using β in (31).

where $\beta_{\text{thresh}} := 2$ is the threshold parameter at which $\beta^{(t)}$ becomes larger than one. We remark that $\beta^{(t)}$ changes only every t_{eig} iterations.

The $\mathbf{D}$-update is multiplicative as in Ostermeier et al. (1994) or Krause and Glasmachers (2015) to be unbiased on the log-scale and to simply guarantee the positive definiteness of $\mathbf{D}$. Note that the first order approximation of the update formula $\mathbf{D}^{(t)} \exp((2\beta)^{-1}\Delta_D)$ gives

$$\left(\mathbf{D}^{(t+1)}\right)^2 \approx \left(\mathbf{D}^{(t)}\right)^2 + \frac{1}{\beta^{(t)}}\mathbf{D}^{(t)}\Delta_D\mathbf{D}^{(t)} \ ,$$

which is the update in separable CMA.

Importantly, we set the learning rates for the $\mathbf{D}$ update, $c_{1,D}$ and $c_{\mu,D}$, to be about n times greater than the learning rates for the $\mathbf{C}$ update, c_1 and c_μ. Moreover, we maintain an additional evolution path, $\boldsymbol{p}_{c,D}$, for the rank-one update of $\mathbf{D}$ since an adequate cumulation factor, $c_{c,D}$, may be different from the one for $\boldsymbol{p}_c$.

4.2 Damping factor

The *dynamic* damping factor $\beta^{(t)}$ is the crucial novelty that prevents diagonal decoding to disrupt the adaptation of the covariance matrix $\mathbf{C}$ in CMA-ES. The damping factor is introduced to prevent the sampling distributions from drastically changing due to the diagonal matrix update. Figure 3 visualizes three example cases with different $\mathbf{C}$. When the diagonal decoding matrix changes from the identity matrix to $\mathbf{D}$, the change of the distribution from $\mathcal{N}(\mathbf{0}, \mathbf{C})$ to $\mathcal{N}(\mathbf{0}, \mathbf{DCD})$ is minimal when $\mathbf{C}$ is diagonal, and is comparatively large if $\mathbf{C}$ is non-diagonal. It will be greater as the condition number of the correlation matrix $\mathrm{corr}(\mathbf{C})$ increases. In the worst case in Figure 3, we see that two distributions are overlapping only at the center of distribution.

The damping factor $\beta^{(t)}$ computed in (31) is motivated from the KL divergence between the sampling distributions before and after the $\mathbf{D}$ update. The KL divergence between two multivariate normal distributions that have the same mean vectors is equivalent to the half of the Log-Determinant divergence between the covariance matrices, defined as

$$D_{ld}(\mathbf{A}, \mathbf{B}) = \mathrm{Tr}(\mathbf{AB}^{-1}) - \log\det(\mathbf{AB}^{-1}) - n \ , \tag{32}$$

where $\mathbf{A}$ and $\mathbf{B}$ are two positive definite symmetric matrices. Let $\mathbf{A} = \mathbf{D}'\mathbf{C}\mathbf{D}'$ and $\mathbf{B} = \mathbf{D}\mathbf{C}\mathbf{D}$, i.e., $\mathbf{A}$ and $\mathbf{B}$ are the covariance matrices after and before the $\mathbf{D}$ update, respectively. The divergence D_{ld} is asymmetric, and the value changes if we exchange $\mathbf{A}$ and $\mathbf{B}$, but we will obtain a symmetric approximation in the end. To analyze the divergence between $\mathbf{A}$ and $\mathbf{B}$ due to the difference in shape and put aside the effect of a global scaling, we scale both to be of determinant one. We denote the resulting divergence by $\bar{D}_{ld}(\mathbf{A}, \mathbf{B}) = D_{ld}(\mathbf{A}/\det(\mathbf{A}), \mathbf{B}/\det(\mathbf{B}))$. Then, we obtain

$$\bar{D}_{ld}(\mathbf{A}, \mathbf{B}) = \mathrm{Tr}\left(\exp\left(\frac{1}{2\beta}\bar{\Delta}_D\right)\mathbf{C}\exp\left(\frac{1}{2\beta}\bar{\Delta}_D\right)\mathbf{C}^{-1}\right) - n \ ,$$

where $\bar{\Delta}_D = \Delta_D - (\mathrm{Tr}(\Delta_D)/n)\mathbf{I}$. With the approximation $\exp\left(\frac{1}{2\beta}\bar{\Delta}_D\right) \approx \mathbf{I} + \left(\frac{1}{2\beta}\right)\bar{\Delta}_D + \frac{1}{2}\left(\frac{1}{2\beta}\right)^2\bar{\Delta}_D^2$ and neglecting $(\Delta_D/\beta)^3$ and higher terms, we have[10]

$$\begin{aligned}\bar{D}_{ld}(\mathbf{A}, \mathbf{B}) &\approx \left(\frac{1}{2\beta}\right)^2 \mathrm{Tr}\left(\bar{\Delta}_D\mathbf{C}\bar{\Delta}_D\mathbf{C}^{-1}\right) + \left(\frac{1}{2\beta}\right)^2 \mathrm{Tr}\left(\bar{\Delta}_D^2\right) \\ &\leqslant \frac{(\beta + \beta_{\mathrm{thresh}} - 1)^2 + 1}{4\beta^2}\,\mathrm{Tr}\left(\bar{\Delta}_D^2\right) \ .\end{aligned}$$

Note that $\bar{D}_{ld}((\mathbf{D}^{(t+1)})^2, (\mathbf{D}^{(t)})^2) \approx 2^{-1}\,\mathrm{Tr}\left(\bar{\Delta}_D^2\right)$ if $\mathbf{C}$ is diagonal and $\beta = 1$. Therefore, with β computed in (31), we bound the realized divergence approximately by the divergence of $\mathbf{D}$ like

$$\bar{D}_{ld}(\mathbf{D}^{(t+1)}\mathbf{C}\mathbf{D}^{(t+1)}, \mathbf{D}^{(t)}\mathbf{C}\mathbf{D}^{(t)}) \leqslant \bar{D}_{ld}((\mathbf{D}^{(t+1)})^2, (\mathbf{D}^{(t)})^2) \tag{33}$$

for large β.

In a nutshell, we have quantified the distribution change from changing $\mathbf{D}$ by measuring its KL-divergence. We derive that β in (31) upper bounds the KL-divergence due to changes of $\mathbf{D}$ approximately by the KL-divergence from the same change of $\mathbf{D}$ when $\mathbf{C} = \mathbf{I}$. The latter is directly determined by the learning rates $c_{1,D}$ and $c_{\mu,D}$.

4.3 Implementation Remark

To avoid unnecessary numerical errors and additional computational effort for eigen decompositions in the adaptive diagonal decoding mechanism (28), (29), (31), we force the diagonal elements of $\mathbf{C}$ to be all one by assigning

$$\mathbf{D}^{(t)} \leftarrow \mathbf{D}^{(t)}\,\mathrm{diag}(\mathbf{C}^{(t)})^{\frac{1}{2}} \tag{34}$$

$$\mathbf{C}^{(t)} \leftarrow \mathrm{corr}(\mathbf{C}^{(t)}) = \mathrm{diag}(\mathbf{C}^{(t)})^{-\frac{1}{2}}\mathbf{C}^{(t)}\,\mathrm{diag}(\mathbf{C}^{(t)})^{-\frac{1}{2}} \ . \tag{35}$$

These lines are performed just before the eigen decomposition. It means that $\mathbf{D}$ and $\mathbf{C}$ are the variance matrix and the correlation matrix of the sampling distribution, respectively. This reparametrization does not change the algorithm itself, it only improves

[10] Let $\mathbf{A}$ and $\mathbf{B}$ be an arbitrary positive definite symmetric matrix and an arbitrary symmetric matrix, respectively. Let $\otimes$ and vec denote the Kronecker product of two matrices and the matrix-vector rearrangement operator that successively stacks the columns of a matrix. From Theorem 16.2.2 of Harville (2008), we have $\mathrm{Tr}(\mathbf{A}\mathbf{B}\mathbf{A}^{-1}\mathbf{B}) = \mathrm{vec}(\mathbf{B})^{\mathsf{T}}(\mathbf{A}\otimes\mathbf{A}^{-1})\mathrm{vec}(\mathbf{B})$. The eigenvalues of the Kronecker product $\mathbf{A}\otimes\mathbf{A}^{-1}$ are the products of the eigenvalues of $\mathbf{A}$ and $\mathbf{A}^{-1}$ (Theorem 21.11.1 of Harville (2008)). They are all positive and upper bounded by the condition number of $\mathbf{A}$. Therefore, we have $\mathrm{vec}(\mathbf{B})^{\mathsf{T}}(\mathbf{A}\otimes\mathbf{A}^{-1})\mathrm{vec}(\mathbf{B}) \leqslant \mathrm{Cond}(\mathbf{A})\|\mathrm{vec}(\mathbf{B})\|^2 = \mathrm{Cond}(\mathbf{A})\,\mathrm{Tr}(\mathbf{B}^2)$. Letting $\mathbf{A} = \mathrm{corr}(\mathbf{C})$ and $\mathbf{B} = \bar{\Delta}_D$, we have $\mathrm{Tr}\left(\bar{\Delta}_D\mathbf{C}\bar{\Delta}_D\mathbf{C}^{-1}\right) = \mathrm{Tr}(\mathbf{A}\mathbf{B}\mathbf{A}^{-1}\mathbf{B}) \leqslant (\beta + \beta_{\mathrm{thresh}} - 1)^2\,\mathrm{Tr}(\bar{\Delta}_D^2)$.

the numerical stability of the implementation. Note that if $\mathbf{C}$ is constrained to be a correlation matrix, then $\mathbf{DCD}$ is a *unique* parametrization for the covariance matrix.

The eigen decomposition of $\mathbf{C}^{(t)}$ is performed every t_{eig} iterations. We keep $\sqrt{\mathbf{C}}$ and $\sqrt{\mathbf{C}}^{-1}$ until the next matrix decomposition is performed. Suppose that the eigen decomposition $\mathbf{C}^{(t)} = \mathbf{E}\Lambda\mathbf{E}^\mathsf{T}$ is performed at iteration t. Then, the above matrices are $\sqrt{\mathbf{C}} = \mathbf{E}\Lambda^{\frac{1}{2}}\mathbf{E}^\mathsf{T}$ and $\sqrt{\mathbf{C}}^{-1} = \mathbf{E}\Lambda^{-\frac{1}{2}}\mathbf{E}^\mathsf{T}$. Then, we can compute β in (31) as $\beta = \max(1, (\max_i([\Lambda]_{i,i})/\min_i([\Lambda]_{i,i}))^{\frac{1}{2}} - \beta_{\text{thresh}} + 1)$. This β is kept until the next eigen decomposition is performed. Then, the additional computational cost for adaptive diagonal decoding is $O(n^2/t_{\text{eig}} + \lambda n)$, which is smaller than the computational cost for the other parts of the CMA-ES, $O(n^3/t_{\text{eig}} + \lambda n^2)$ per iteration.

One might think that maintaining $\mathbf{D}$ is not necessary and $\mathbf{C}$ could be updated by pre- and post-multiplying by a diagonal matrix absorbing the effect of the $\mathbf{D}$ update. However, because diagonal decoding may lead to a fast change of the distribution shape, the decomposition of $\mathbf{C}$ then needs to be done every iteration. The chosen parametrization circumvents frequent matrix decompositions despite changing the sampling distribution rapidly.

5 Algorithm Summary and Strategy Parameters

The dd-CMA-ES combines weighted recombination, active covariance matrix update with positive definiteness guarantee (described in Section 3.1), adaptive diagonal decoding (described in Section 4) and CSA, and is summarized in Algorithm 1.[11]

Providing good default values for the strategy parameters (aka hyper parameters) is, needless to say, essential for the success of a novel algorithm. Especially in a black-box optimization scenario, parameter tuning for an application relies on trial-and-error and is usually prohibitively time consuming. The computation of the default parameter values is summarized in Algorithm 2. Since we have $c_1/c_\mu = c_{1,D}/c_{\mu,D}$ as long as $c_1 + c_\mu < 1$ (see below), the recombination weights for the update of $\mathbf{C}$ and of $\mathbf{D}$ are the same, $w_i = w_{i,D}$.

The learning rates for $\mathbf{D}$ and $\mathbf{C}$ are modified to improve the adaptation speed especially in high dimension. Let m be the degrees of freedom of the matrix to be adapted, i.e., $m = (n+1)n/2$ for the $\mathbf{C}$-update and $m = n$ for the $\mathbf{D}$-update. The learning rate parameters and the cumulation factors are set to the following values

$$c_1,\ c_{1,D} = \frac{1}{2(m/n+1)(n+1)^{3/4} + \mu_w/2} \tag{36}$$

$$c_\mu,\ c_{\mu,D} = \min(\mu' c_1, 1-c_1),\ \min(\mu' c_{1,D}, 1-c_{1,D}) \tag{37}$$

$$\text{with } \mu' = \mu_w + \frac{1}{\mu_w} - 2 + \frac{\lambda}{2(\lambda+5)}$$

$$c_c,\ c_{c,D} = \frac{\sqrt{\mu_w c_1}}{2},\ \frac{\sqrt{\mu_w c_{1,D}}}{2}\ . \tag{38}$$

The first important change is the scaling of the learning rate with the dimension n. In previous works (Ros and Hansen, 2008; Hansen and Auger, 2014; Akimoto and Hansen, 2016), the default learning rates were set inversely proportional to m, i.e., $\Theta(n^{-2})$ for the original CMA and $\Theta(n^{-1})$ for the separable CMA. Our choice is based

[11] Its python code is available at
`https://gist.github.com/youheiakimoto/1180b67b5a0b1265c204cba991fa8518`

Algorithm 1 dd-CMA-ES: CMA-ES with adaptive diagonal decoding

Require: $\boldsymbol{m}$, σ ▷ initial distribution parameters

$\mathbf{D} = \sqrt{\mathbf{C}} = \sqrt{\mathbf{C}}^{-1} = \mathbf{I}$, $\mathbf{K} = \mathbf{O}$ ($\mathbf{O}$: zero matrix) , $\boldsymbol{p}_c = \boldsymbol{p}_\sigma = \boldsymbol{p}_{c,D} = \mathbf{0}$, $\gamma_c = \gamma_\sigma = 0$, $\beta = 1$, $t = 0$
repeat
  sample λ points $(\boldsymbol{z}_i,\ \boldsymbol{y}_i,\ \boldsymbol{x}_i)_{i=1}^{\lambda}$ by (1)
  evaluate $f(\boldsymbol{x}_i)$ for all $i = 1, \ldots, \lambda$
  sort in ascending order of $f(\boldsymbol{x}_i)$ and denote ith best point as $\boldsymbol{z}_{i:\lambda}$, $\boldsymbol{y}_{i:\lambda}$, and $\boldsymbol{x}_{i:\lambda}$
  update $\boldsymbol{m}$ by (2)
  update $\boldsymbol{p}_\sigma$, γ_σ and σ by (3), (4) and (5), then compute h_σ by (11)
  update $\boldsymbol{p}_c$ and $\boldsymbol{p}_{c,D}$ by (9) with factors c_c and $c_{c,D}$, resp., and update γ_c by (10)
  compute $\mathbf{Z}^{(t)}$ by (19) and add it to $\mathbf{K}$ ▷ $\mathbf{K} = \sum_{k=t-(t \mod t_{\text{eig}})}^{t} \mathbf{Z}^{(k)}$
  update $\mathbf{D}$ by (28) and (29) with $\beta^{(t)} = \beta$
  $t \leftarrow t + 1$
  if $(t \mod t_{\text{eig}}) = 0$ **then**
    compute $\mathbf{C}$ by (21) and α by (22) using $\mathbf{K}$
    $\mathbf{D} \leftarrow \mathbf{D}\sqrt{\operatorname{diag}(\mathbf{C})}$ and $\mathbf{C} \leftarrow \sqrt{\operatorname{diag}(\mathbf{C})}^{-1}\mathbf{C}\sqrt{\operatorname{diag}(\mathbf{C})}^{-1}$ ▷ (34) and (35)
    perform eigen decomposition $\mathbf{C} = \mathbf{E}\Lambda\mathbf{E}^{\mathsf{T}}$
    $\beta \leftarrow \max(1, \sqrt{\max_i [\Lambda]_{i,i} / \min_j [\Lambda]_{j,j}} - \beta_{\text{thresh}} + 1)$ ▷ (31)
    $\sqrt{\mathbf{C}} \leftarrow \mathbf{E}\Lambda^{\frac{1}{2}}\mathbf{E}^{\mathsf{T}}$, $\sqrt{\mathbf{C}}^{-1} \leftarrow \mathbf{E}\Lambda^{-\frac{1}{2}}\mathbf{E}^{\mathsf{T}}$, $\mathbf{K} \leftarrow \mathbf{O}$
  end if
until termination condition are met

on empirical observations that $\Theta(n^{-2})$ is exceedingly conservative for higher dimensional problems, say $n > 100$: in experiments to identify c_μ for the original CMA-ES, we vary c_μ and investigate the number of evaluations to reach a given target. We find that the location of the minimum and the shape of the dependency remains for a wide range of different dimensions roughly the same when the evaluations are taken against $c_\mu m / n^{0.35}$ (see also Figure 5 below). The observation suggests in particular that $c_\mu = \Theta(\sqrt{n}/m)$ is likely to fail with increasing dimension, whereas $c_\mu = \Theta(n^{0.35}/m)$ is a stable setting such that the new setting of $\Theta(n^{0.25}/m)$ (replacing $\Theta(n^0/m)$) is still sufficiently conservative. Figure 4 depicts the difference between the default learning rate values in Hansen and Auger (2014) and the above formulas.[12]

Another important change in the default parameter values from previous studies (Ros and Hansen, 2008; Hansen and Auger, 2014) is in the cumulation factor c_c. Previously, the typical choices were $c_c = \frac{4}{n+4}$ or $c_c = \frac{4+\mu_w/n}{n+4+2\mu_w/n}$. The new and simpler default value for the cumulation factor[13] is motivated by an experiment on the Cigar

[12] The learning rate for the rank-μ update is usually μ' times greater than the learning rate for the rank-one update, where $\mu' \in (\mu_w - 2, \mu_w - 1/2)$ is monotonous in μ_w and approaches $\mu_w - 3/2$ for $\lambda \to \infty$. Using μ' instead of μ_w is a correction for small μ_w. When $\mu_w = 1$, we have $\mu' < 1/2$ (the first three terms cancel out). In this case, because $\mu = 1$, the rank-one update contains more information than the rank-μ update as the evolution path accumulates information over time. Using μ_w instead of μ' would result in the same learning rates for both updates, whereas the learning rate for the rank-μ update should be smaller in this case.

[13] The appearance of $\sqrt{\mu_w}$ in the cumulation factor is motivated as follows. Hansen and Ostermeier (2001) analyzed the situation where the selection vector $\sqrt{\mu_w}\sum_{i=1}^{\mu} w_i \boldsymbol{z}_{i:\lambda}$ alternates between two vectors $\boldsymbol{v}$ and $\boldsymbol{w}$ over iterations and showed the squared Euclidean norm of the evolution path, i.e. the eigenvalue of $\boldsymbol{p}_c\boldsymbol{p}_c^{\mathsf{T}}$, remains in $O(\|\boldsymbol{v} + \boldsymbol{w}\|^2/c_c)$ if $\|\boldsymbol{v} + \boldsymbol{w}\| > 0$. On the other hand, Akimoto et al. (2008) showed that the above selection vector can potentially be proportional to $\sqrt{\mu_w}$ when σ is (too) small. Altogether, we have

Algorithm 2 Default parameter computation for dd-CMA-ES

Require: n and optionally λ

$\lambda = 4 + \lfloor 3 \ln n \rfloor$ if not given

$w'_i = \ln \frac{\lambda+1}{2} - \ln i$ for $i = 1, \ldots, \lambda$

$\mu_w = \frac{(\sum_{i:w'_i>0} |w'_i|)^2}{\sum_{i:w'_i>0} |w'_i|^2}$ and $\mu_w^- = \frac{(\sum_{i:w'_i<0} |w'_i|)^2}{\sum_{i:w'_i<0} |w'_i|^2}$

$c_m = 1$

$c_\sigma = \frac{\mu_w+2}{n+\mu_w+5}$ and $d_\sigma = 1 + c_\sigma + 2 \max\left(0, \sqrt{\frac{\mu_w-1}{n+1}} - 1\right)$

$c_1,\ c_{1,D} = \frac{1}{2(m/n+1)(n+1)^{3/4}+\mu_w/2}$ with $m = n(n+1)/2$ and n, resp.

$c_\mu,\ c_{\mu,D} = \min(\mu' c_1, 1 - c_1),\ \min(\mu' c_{1,D}, 1 - c_{1,D})$ with $\mu' = \mu_w + \frac{1}{\mu_w} - 2 + \frac{1}{2}\frac{\lambda}{\lambda+5}$

$c_c,\ c_{c,D} = \frac{\sqrt{\mu_w c_1}}{2}, \frac{\sqrt{\mu_w c_{1,D}}}{2}$

$$w_i, w_{i,D} = \begin{cases} \frac{w'_i}{\sum_{j:w'_j>0} |w'_j|} & \text{for } w'_i \geqslant 0 \\ \frac{w'_i \times \min\left(1+r,\ 1+2\mu_w^-/(\mu_w+2)\right)}{\sum_{j:w'_j<0} |w'_j|} & \text{for } w'_i < 0 \end{cases} \quad \text{with } r = \frac{c_1}{c_\mu} \text{ and } \frac{c_{1,D}}{c_{\mu,D}}, \text{ resp.}$$

$t_{\text{eig}} = \max\left(1, \lfloor (\beta_{\text{eig}}(c_1 + c_\mu))^{-1} \rfloor\right)$ with $\beta_{\text{eig}} = 10n$

$\beta_{\text{thresh}} = 2$

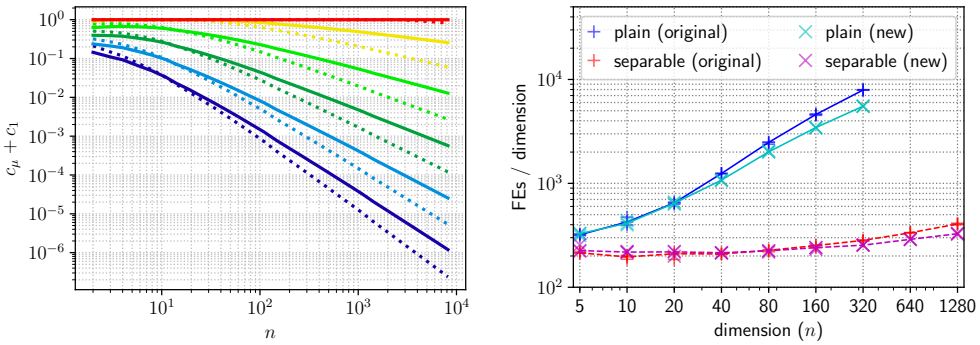
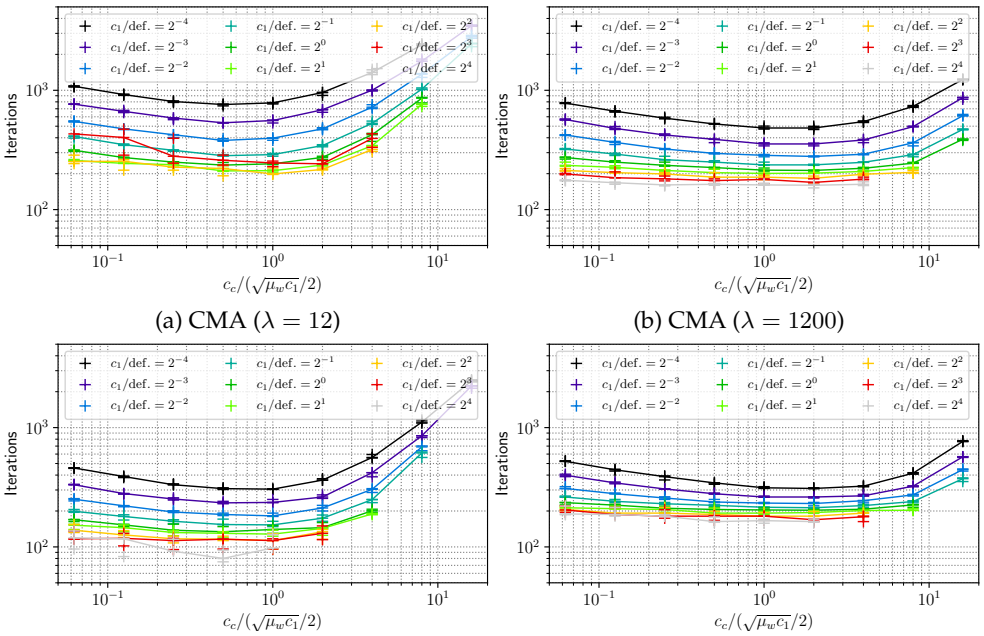


Figure 4: Left: Scaling of $c_1 + c_\mu$ with the dimension n for different population sizes. Solid lines are for the new setting from Table 2, $\Theta(n^{1/4}\lambda/m)$ and dashed lines are for the original setting, $\Theta(\lambda/m)$. For $n < 10$, the new learning rates are slightly more conservative, for $n > 20$ they are more ambitious. Shown are six population sizes equally log-spaced between $\lambda = 4 + \lfloor 3 \log(n) \rfloor$ (bottom) and $\lambda = 64n^2$ (top). Right: Function evaluations on the ellipsoid function for plain and separable CMA with original and new parameter settings. Three independent trials have been conducted for each setting, and the median of each setting is indicated by a line (solid: plain CMA, dashed: separable CMA). See Section 6 for details.

function, investigating the dependency of c_c on c_1. The effect of the rank-one update of the covariance matrix $\mathbf{C}$ is most pronounced when the covariance matrix needs to increase a single eigenvalue. To skip over the usual initial phase where sigma decreases without changing the covariance matrix and emphasize on the effect of c_c, the mean

$c_1 \|\boldsymbol{p}_c\|^2 \in O(c_1 \mu_w / c_c)$, where $c_1 \mu_w \to 2$ and $c_c \to \sqrt{2}/2$ as $\mu_w \to \infty$. Therefore, the eigenvalue added by the rank-one update is bounded by a constant with overwhelming probability.



(a) CMA ($\lambda = 12$)

(b) CMA ($\lambda = 1200$)

(c) SEP ($\lambda = 12$)

(d) SEP ($\lambda = 1200$)

Figure 5: Function evaluations on the 20-D cigar function until the target f-value of 10^{-2} is achieved for different c_1 (or $c_{1,D}$) and different c_c (or $c_{c,D}$). Three independent trials have been conducted for each setting, and the median of each setting is indicated by a line. The label *def* is the default value computed in (36). Missing points indicate more than 5000 iterations or invalid c-values, e.g., $c_1 > 1$ or $c_c > 1$.

vector is initialized at $\boldsymbol{m}^{(0)} = (1, 0, \ldots, 0)$ and $\sigma^{(0)} = 10^{-6}$. The rank-μ update is off, i.e., $c_\mu = c_{\mu,D} = 0$. We compared the number of f-calls until the target function value of 10^{-2} is reached. Note that $\sigma^{(0)}$ is chosen to avoid σ adaptation at the beginning and the target value is chosen so that we stop the optimization once the covariance matrix learned the right scaling. Figure 5 compares the number of f-calls spent by the original CMA and the separable CMA on the 20 dimensional Cigar function. Because the x-axis is normalized with the new default value for c_c, the shape of the graph and the location of the minimum remains roughly the same for different values of c_1 and λ. The adaptation speed of the covariance matrix tends to be the best around $c_c = \sqrt{\mu_w c_1}/2$ or $\sqrt{\mu_w c_{1,D}}/2$. The minimum is more pronounced for smaller c_1 (or $c_{1,D}$). Nevertheless, we observe little improvement with the new setting in practice since more f-calls are usually spent to adapt the overall covariance matrix and even to adapt the step-size to converge. We have done the same experiments in dimension 80 and observe the same trend.

None of the strategy parameters are meant to be tuned by users, except the population size λ. A larger population size typically results in finding a better local minimum on rugged functions such as multimodal or noisy functions (Hansen and Kern, 2004). It is also advantageous when the objective function values are evaluated in parallel (Hansen et al., 2003). Restart strategies that perform independent restarts with increasing population size such as IPOP strategy (Harik and Lobo, 1999; Auger and Hansen,

Table 1: Test function definitions and initial conditions. The unit vector $\boldsymbol{u}$ is either $\boldsymbol{e}_1 = (1, 0, \ldots, 0)$ for separable scenarios, or random vectors drawn uniformly on the unit sphere for non-separable scenarios or for Ell-Cig and Ell-Dis. A vector $\boldsymbol{z}$ represents an orthogonal transformation $\mathbf{R}\boldsymbol{x}$ of the input vector $\boldsymbol{x}$, where the orthogonal matrix $\mathbf{R}$ is the identity matrix for separable scenarios, or is constructed by generating normal random elements and applying the Gram-Schmidt procedure. The diagonal matrix $\mathbf{D}_{\mathrm{ell}} = \mathrm{diag}(1, \cdots, 10^{\frac{i-1}{n-1}}, \cdots, 10)$ represents a coordinate-wise scaling and the vector $\mathbf{y}$ is a coordinate-wisely transformed input vector $\boldsymbol{y} = \mathbf{D}_{\mathrm{ell}}^2 \boldsymbol{x}$.

	$f(\boldsymbol{x})$	$\boldsymbol{m}^{(0)}$	$\sigma^{(0)}$
Sphere	$\|\boldsymbol{x}\|^2$	$3 \cdot \mathbf{1}$	1
Cigar	$\langle \boldsymbol{u}, \boldsymbol{x} \rangle^2 + 10^6 (\|\boldsymbol{x}\|^2 - \langle \boldsymbol{u}, \boldsymbol{x} \rangle^2)$	$3 \cdot \mathbf{1}$	1
Discus	$10^6 \langle \boldsymbol{u}, \boldsymbol{x} \rangle^2 + (\|\boldsymbol{x}\|^2 - \langle \boldsymbol{u}, \boldsymbol{x} \rangle^2)$	$3 \cdot \mathbf{1}$	1
Ellipsoid	$\|\mathbf{D}_{\mathrm{ell}}^3 \boldsymbol{z}\|^2$	$3 \cdot \mathbf{1}$	1
TwoAxes	$10^6 \sum_{i=1}^{n/2} [\boldsymbol{z}]_i^2 + \sum_{i=n/2+1}^{n} [\boldsymbol{z}]_i^2$	$3 \cdot \mathbf{1}$	1
Ell-Cig	$10^{-4} \langle \boldsymbol{u}, \boldsymbol{y} \rangle^2 + (\|\boldsymbol{y}\|^2 - \langle \boldsymbol{u}, \boldsymbol{y} \rangle^2)$	$3 \cdot \mathbf{1}$	1
Ell-Dis	$10^4 \langle \boldsymbol{u}, \boldsymbol{y} \rangle^2 + (\|\boldsymbol{y}\|^2 - \langle \boldsymbol{u}, \boldsymbol{y} \rangle^2)$	$3 \cdot \mathbf{1}$	1
Rosenbrock	$\sum_{i=1}^{n-1} 100([\boldsymbol{z}]_i^2 - [\boldsymbol{z}]_{i+1})^2 + ([\boldsymbol{z}]_i - 1)^2$	$\mathbf{0}$	0.1
Bohachevsky	$\sum_{i=1}^{n-1} ([\boldsymbol{z}]_i^2 + 2[\boldsymbol{z}]_{i+1}^2 - 0.3\cos(3\pi[\boldsymbol{z}]_i) \ldots - 0.4\cos(4\pi[\boldsymbol{z}]_{i+1}) + 0.7)$	$\mathcal{N}(\mathbf{0}, 8^2\mathbf{I})$	7
Rastrigin	$\sum_{i=1}^{n} [\boldsymbol{z}]_i^2 + 10(1 - \cos(2\pi[\boldsymbol{z}]_i))$	$\mathcal{N}(\mathbf{0}, 3^2\mathbf{I})$	2

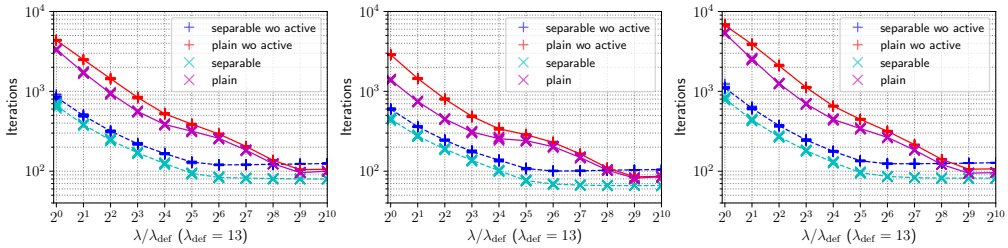
2005b) or BIPOP strategy (Hansen, 2009) or NIPOP strategy (Loshchilov et al., 2012) are handy policies that automate the parameter tuning. They can be applied to the CMA-ES with diagonal acceleration in a straight-forward way. We leave research in this line as future work.

6 Experiments

We conduct numerical simulations to see the effect of the algorithmic components of CMA-ES with diagonal acceleration, dd-CMA-ES—namely the active update with positive definiteness guarantee and the adaptive diagonal decoding. Since the effect of the active covariance matrix update with default population size has been investigated by Jastrebski and Arnold (2006) and our contribution is a positive definiteness guarantee for the covariance matrix especially for large population size, we investigate how the effect of the active update scales as the population size increases. Our main focus is however on the effect of adaptive diagonal decoding. Particularly, we investigate (i) how much better dd-CMA scales on separable functions than the CMA without diagonal decoding (plain CMA), (ii) how closely dd-CMA matches the performance of separable CMA on separable functions, and (iii) how the scaling of dd-CMA compares to the plain CMA on various non-separable functions. Moreover, we investigated how effective dd-CMA is when the population size is increased.

6.1 Common Setting

Table 1 summarizes the test functions used in the following experiments together with the initial $\boldsymbol{m}^{(0)}$ and $\sigma^{(0)}$. The initial covariance matrix $\mathbf{C}^{(0)}$ and the diagonal decoding matrix $\mathbf{D}^{(0)}$ are always set to the identity matrix. The random vectors and the random matrix appearing in Table 1 are initialized randomly for each problem instance, but the



(a) Ellipsoid (b) Discus (c) TwoAxes

Figure 6: Number of iterations spent by plain and separable CMA with and without active update to reach the target f-value of 10^{-8} on the 40 dimensional separable Ellipsoid, Discus, and TwoAxes functions. Each line indicates the median of each setting (solid: plain CMA, dashed: separable CMA).

same values are used for different algorithms for fair comparison. A trial is considered as success if the target function value of 10^{-8} is reached before the algorithm spends $5 \times 10^4 n$ function evaluations, otherwise regarded as failure. For $n \leqslant 40$ we conducted 20 independent trials for each setting, 10 for $80 \leqslant n \leqslant 320$, and 3 for $n \geqslant 640$. When the computational effort of evaluating the test function scales worse than linear with the dimension (i.e., on rotated functions) we may omit dimensions larger than 320.

We compare the following CMA variants:

Plain CMA updates $\mathbf{C}$ while $\mathbf{D}$ is kept the identity matrix, with or without active update described in Section 3 (Algorithm 1 without $\mathbf{D}$-update);

Separable CMA updates $\mathbf{D}$ as described in Section 4 while $\mathbf{C}$ is kept the identity matrix, with active update or without active update, where negative recombination weights are set to zero (Algorithm 1 without $\mathbf{C}$-update);

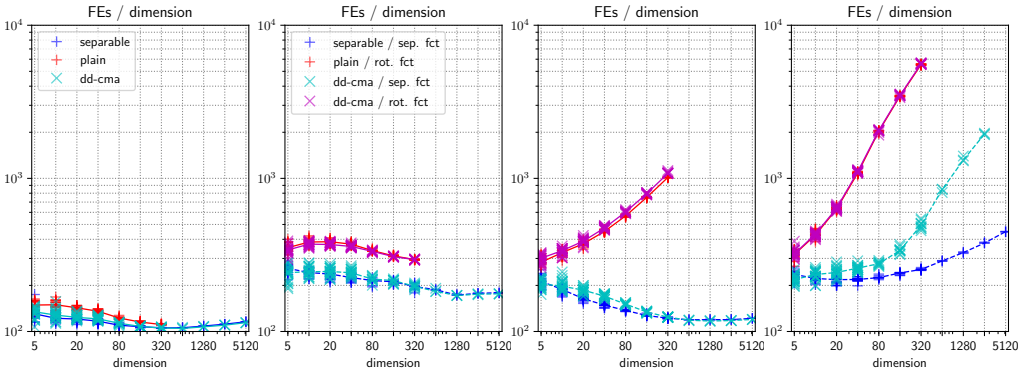
dd-CMA as summarized in Algorithm 1, updates $\mathbf{C}$ as described in Section 2.1 with active update described in Section 4, and updates $\mathbf{D}$ as described in Section 4.

All strategy parameters such as the learning rates are set to their default value presented in Section 5. Note that the separable CMA is different from the original publication (Ros and Hansen, 2008) in that the matrix is updated in multiplicative form which however barely affects the performance.

6.2 Active CMA

First, the effect of the active update is investigated. The plain and separable CMA with and without active update are compared.

Figure 6 shows the number of iterations spent by each algorithm plotted against population size λ. The number of iterations to reach the target function value decreases as λ increases and tends to level out. The plain CMA is consistently faster with active update than without. As expected from the results of Jastrebski and Arnold (2006), the effect of active covariance matrix update is most pronounced on functions with a small number of sensitive variables such as the Discus function. The advantage of the active update diminishes as λ increases in plain CMA, whereas the speed-up in separable CMA becomes even slightly more pronounced.



(a) Sphere (b) Cigar (c) Discus (d) Ellipsoid

Figure 7: Function evaluations (divided by n) spent by each algorithm until it reaches the target f-value of 10^{-8} and their median values (solid line: rotated function; dashed line: non-rotated, separable function). Data for dd-CMA on the separable Ellipsoid in dimension 5120 are missing since they did not finish in a limited CPU time.

6.3 Adaptive Diagonal Decoding

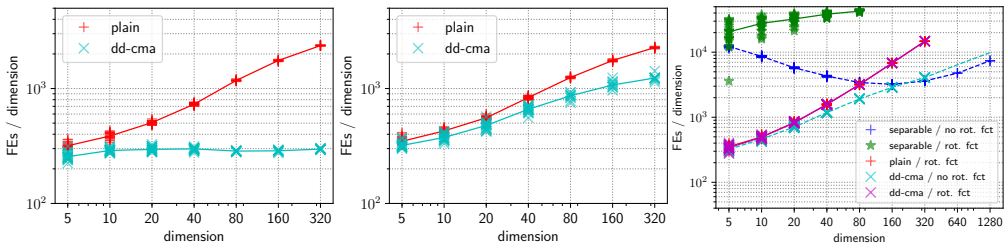
Next, we compare dd-CMA with the plain and the separable CMA with active update.

Figure 7 compares dd-CMA with the plain CMA on the rotated Cigar, Ellipsoid, and Discus functions, and with the separable CMA on the separable Cigar, Ellipsoid, and Discus functions. Note that the plain CMA scales equally well on separable functions and rotated functions, while the separable CMA will not find the target function value on rotated functions before the maximum budget is exhausted (Ros and Hansen, 2008). Displayed are the number of function evaluations to reach the target function value on each problem instance. The results of dd-, plain and separable CMA on the Sphere function are displayed for reference.

On the Sphere function, no covariance matrix adaptation is required, and all algorithms perform quite similarly. On the Cigar function, the rank-one covariance matrix update is known to be rather effective, and even the plain CMA scales linearly on the rotated Cigar function. On both, separable and rotated Cigar functions, dd-CMA is competitive with separable and plain CMA thereby combining the better performance of the two.

The discrepancy between plain and separable CMA is much more pronounced on Ellipsoid and Discus functions, where the plain CMA scales super-linearly, whereas the separable CMA exhibits linear or slightly worse than linear scaling on separable instances and fails to find the optimum within the given budget on rotated instances. On the rotated functions, dd-CMA is competitive with the plain CMA, whereas it significantly reduces the number of function evaluations on the separable functions compared to plain CMA, e.g., ten times less f-calls are required on the 160 dimensional separable Ellipsoid function.

The dd-CMA is competitive with separable CMA on the separable Discus function up to 5120 dimension, which is the ideal situation since we can not expect faster adaptation of the distribution shape on separable functions. On the separable Ellipsoid function, the performance curve of dd-CMA starts deviating from that of the separable CMA around $n = 320$, and scales more or less the same as the plain CMA afterwards, yet it requires ten times less f-calls. To improve the scaling of dd-CMA on the sepa-



(a) Ell-Cig (b) Ell-Dis (c) Rosenbrock

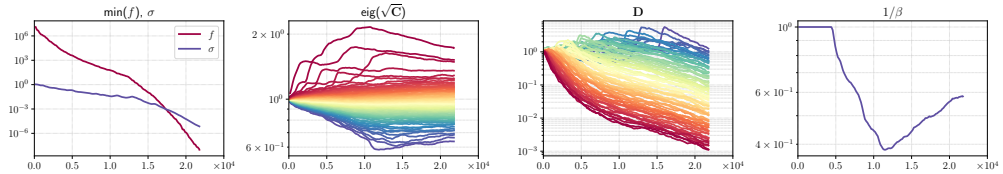
Figure 8: Examples where dd-CMA not only performs on par with the better of plain and separable CMA, but outperforms both at least in some dimension up to $n = 320$ (up to $n = 1280$ only for separable CMA on the non-rotated Rosenbrock function). Function evaluations to reach the target f-value of 10^{-8} divided by n. Each line indicates the median of each setting (solid: rotated function, dashed: non-rotated function). The line for dd-CMA is extrapolated up to $n = 1280$ with the least square log-log-linear regression on the median values.

rable Ellipsoid function, we might need to set β_{thresh} depending on n, or use another monotone transformation of the condition number of the correlation matrix in (31). Performing the eigen decomposition of $\mathbf{C}$ less frequently (i.e., setting β_{eig} smaller) is another possible way to improve the performance of dd-CMA on the separable Ellipsoid function, while compromising the performance on the rotated Ellipsoid function[14].

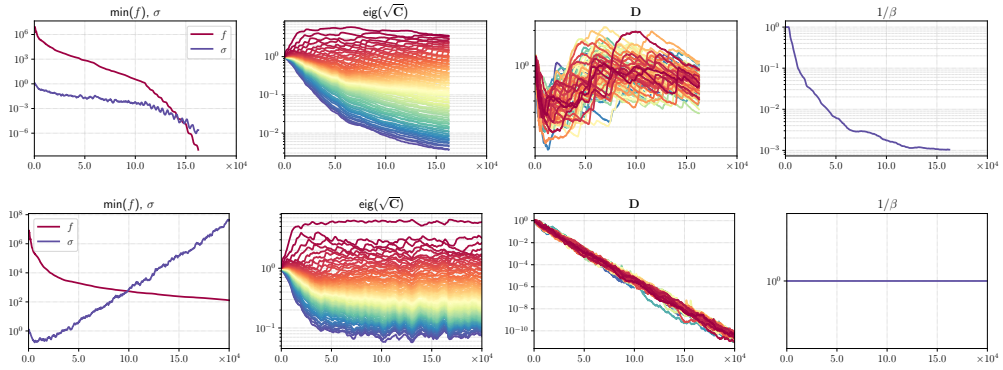
Figures 8a and 8b show the results on Ell-Cig and Ell-Dis functions, which are non-separable ill-conditioned functions with additional coordinate-wise scaling, Figure 8c shows the results on non-rotated and rotated Rosenbrock functions. The separable CMA locates the target f-value within the given budget only on the non-rotated Rosenbrock function in smaller dimension. The experiment on the Ell-Cig function reveals that diagonal decoding can improve the scaling of the plain CMA *even on non-separable functions*. The improvement on the Ell-Dis function is much less pronounced. The reason why, compared to the plain CMA, dd-CMA is more suitable for Ell-Cig than for Ell-Dis is discussed in relation with Figure 9 below. On the non-rotated Rosenbrock function, dd-CMA becomes advantageous as the dimension increases, just as the separable CMA is faster than the plain CMA when $n > 80$.

Figure 9 visualizes insights in the typical behaviour of dd-CMA on different functions. On the separable Ellipsoid, the correlation matrix deviates from the identity matrix due to stochastics and the initial parameter setting, and the inverse damping decreases marginally. This effect is more pronounced for $n > 320$ and is the reason for the impaired scaling of dd-CMA on the separable Ellipsoid in Figure 7d. The diagonal decoding matrix, i.e., variance matrix, adapts the coordinate-wise scaling efficiently as long as the condition number of the correlation matrix $\mathbf{C}$ is not too high. On the rotated Ellipsoid function, where the diagonal elements of the inverse Hessian of the objective function, $\frac{1}{2}\mathbf{R}^{\mathsf{T}}\mathbf{D}_{\mathrm{ell}}^{-6}\mathbf{R}$, likely have similar values, the diagonal decoding matrix remains close to the identity and the correlation matrix learns the ill-conditioning. The inverse damping parameter decreases so that the adaptation of the diagonal decoding matrix

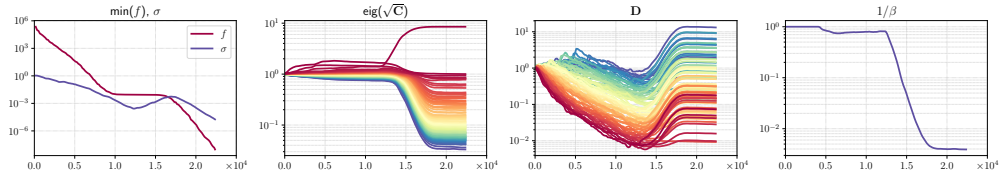
[14] If we set $\beta_{\mathrm{eig}} = 10$ instead of $\beta_{\mathrm{eig}} = 10n$, dd-CMA spends about 1.5 times more FEs on the 5120-dimensional separable Ellipsoid function than separable CMA as displayed in Figure 7d, whereas it spends about 15% more FEs on the 320-dimensional rotated Ellipsoid function than plain CMA as displayed in Figure 7d. This might be a more practical choice, but further investigation is required.



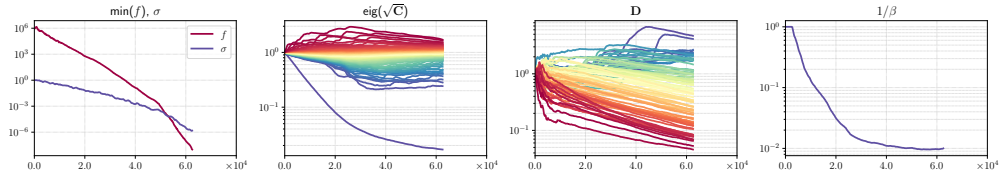
(a) separable Ellipsoid



(b) rotated Ellipsoid, below with fixed $\beta = 1$ instead of (31)



(c) Ellipsoid-Cigar

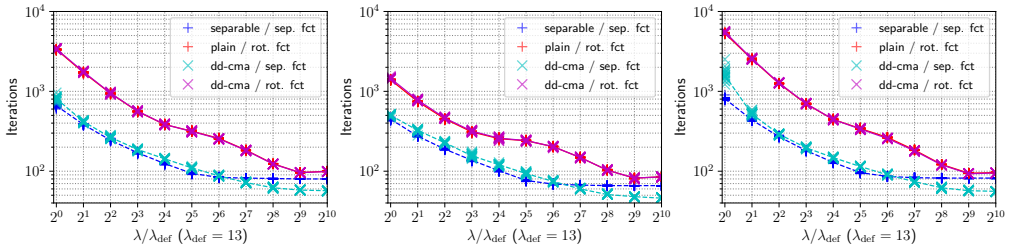


(d) Ellipsoid-Discus

Figure 9: Typical runs of dd-CMA-ES on 80 dimensional test problems (x-axis: function evaluations). Note that the change of $\mathbf{D}$ partially comes from the update of $\mathbf{C}$.

does not disturb the adaptation of the distribution shape. Figure 9b, below, shows that adaptive diagonal decoding without the dynamic damping factor (31) severely disturbs the adaptation of $\mathbf{C}$ on the rotated Ellipsoid.

Ideally, the diagonal decoding matrix $\mathbf{D}$ adapts the coordinate-wise standard deviation and $\mathbf{C}$ adapts the correlation matrix of the sampling distribution. If the correlation matrix $\mathbf{C}$ is easier to adapt than the full covariance matrix $\mathbf{DCD}$, we expect a speed-up of the adaptation of the distribution shape. The functions Ell-Cig and Ell-Dis in Figures 9c and 9d are such examples. Once $\mathbf{D}$ becomes inversely proportional to $\mathbf{D}_{\text{ell}}^2$, the problems become rotated Cigar and rotated Discus functions, respectively.



(a) Ellipsoid (b) Discus (c) TwoAxes

Figure 10: Number of iterations to reach the target f-value of 10^{-8} and its median (solid line: rotated function, dashed line: non-rotated function) versus population size on the 40 dimensional Ellipsoid, Discus, and TwoAxes functions.

The run on the Ell-Cig function in 9c depicts the ideal case. The coordinate-wise scaling is first adapted by $\mathbf{D}$ and then the correlation matrix learns a long Cigar axis. The inverse damping factor is kept at a high value similar to the one observed on the separable Ellipsoid and it starts decreasing only after $\mathbf{D}$ is adapted. On the Ell-Dis function (9d) however, the short non-coordinate axis is learned (too) quickly by the correlation matrix. Therefore, the inverse damping factor decreases before $\mathbf{D}$ has adapted the full coordinate-wise scaling. Since the function value is more sensitive in the subspace corresponding to great eigenvalues of the Hessian of the objective and the ranking of candidate solutions is mostly determined by this subspace, the CMA-ES first shrinks the distribution in this subspace. This is the main reason why dd-CMA is more efficient on Ell-Cig than on Ell-Dis.

Figure 10 shows, similar to Figure 6, the number of iterations versus the population size λ on three functions in dimension 40. For all larger populations sizes up to 13312, dd-CMA performs on par with the better of plain and separable CMA, as ideally to be expected.

We remark that dd-CMA with default $\lambda = 13$ has a relatively large variance in performance on the separable TwoAxes function. Increasing λ from its default value reduces this variance and even reduces the number of f-calls to reach the target. This defect of dd-CMA is even more pronounced for higher dimensions. It may suggest to increase the default λ for dd-CMA. We leave this to be addressed in future work.

Figure 11 shows the number of iterations in successful runs, the success rate and the average number of evaluations in successful runs divided by the success rate (Price, 1997; Auger and Hansen, 2005a) on two multimodal 40-dimensional functions. The maximum numbers of f-calls are $5n \times 10^4$ on Bohachevsky and $2n \times 10^5$ on Rastrigin. Comparing separable to dd-CMA on separable functions and plain to dd-CMA on rotated functions, dd-CMA tends to spend less iterations but to require greater λ to reach the same success rate. The latter is attributed to doubly shrinking the overall variance by updating $\mathbf{C}$ and $\mathbf{D}$. For $\lambda \gg n$, we have $c_\mu \approx 1$ and $c_{\mu,D} \approx 1$ and the effect of shrinking the overall variance due to $\mathbf{C}$ and $\mathbf{D}$ updates is not negligible. Then, the overall variance is smaller than in plain and separable CMA, which also leads to faster convergence. This effect disappears if we force the $\mathbf{D}$ update to keep its determinant unchanged, which can be realized by replacing Δ_D in (29) with $\Delta_D - (\mathrm{Tr}(\Delta_D)/n)\mathbf{I}$.

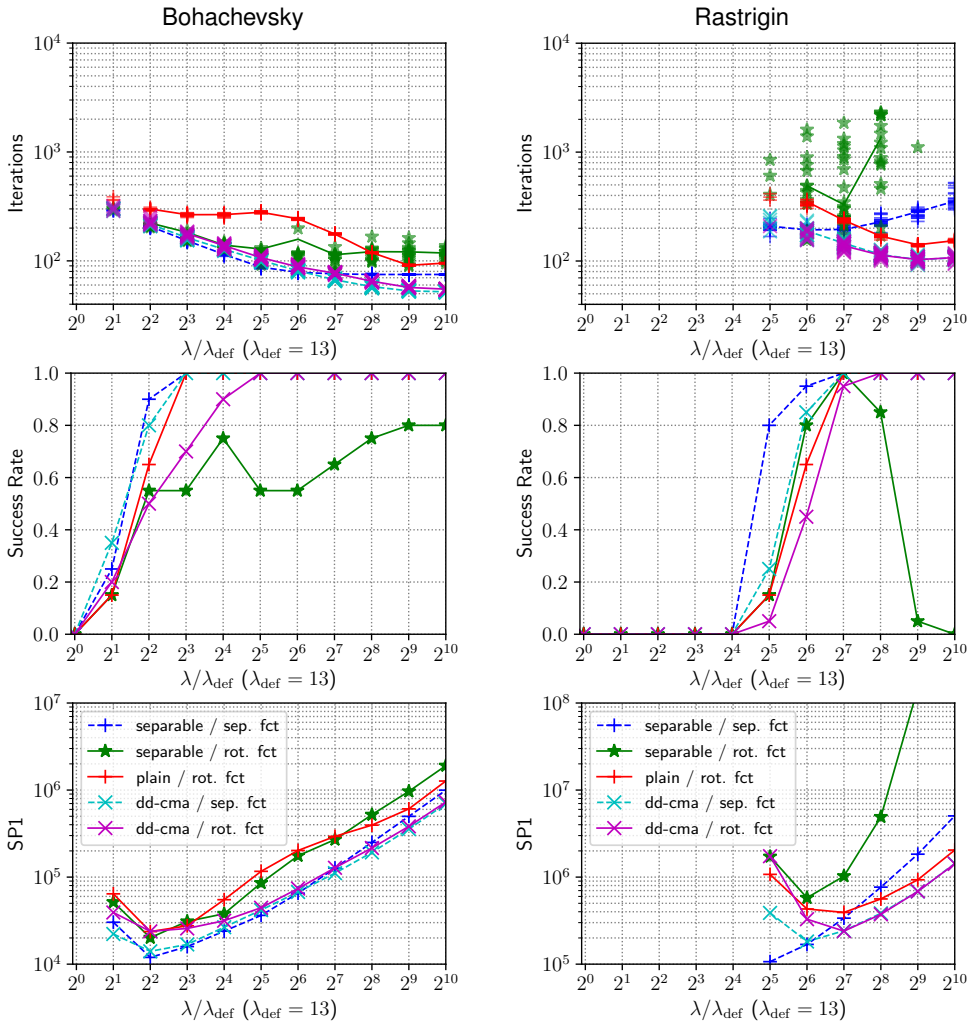


Figure 11: Number of iterations (top) spent by each algorithm until it reaches the target f-value of 10^{-8} on the 40 dimensional Bohachevsky and Rastrigin functions and its median (line), success rates (middle), and the average number of evaluations in successful runs divided by the success rate (SP1, bottom).

7 Summary and Conclusion

In this paper we have put forward several techniques to improve multi-recombinative non-elitistic covariance matrix adaptation evolution strategies without changing their internal computation effort.

The first component is concerned with the active covariance matrix update, which utilizes unsuccessful candidate solutions to actively shrink the sampling distribution in unpromising directions. We propose two ways to guarantee the positive definiteness of the covariance matrix by rescaling unsuccessful candidate solutions.

The second and main component is a diagonal acceleration of CMA by adaptive diagonal decoding, dd-CMA. The covariance matrix adaptation is accelerated by adapt-

ing a coordinate-wise scaling separately from the positive definite symmetric covariance matrix. This drastically accelerates the adaptation speed on high-dimensional functions with coordinate-wise ill-conditioning, whereas it does not significantly influence the adaptation of the covariance matrix on highly ill-conditioned non-separable functions without coordinate-wise scaling component.

The last component is a set of improved default parameters for CMA. The scaling of the learning rates are relaxed from $\Theta(1/n^2)$ to $\Theta(1/n^{1.75})$ for default CMA and from $\Theta(1/n)$ to $\Theta(1/n^{0.75})$ for separable CMA. This contributes to accelerate the learning in all investigated CMA variants on high dimensional problems, say $n \geqslant 100$.

Algorithm selection as well as hyper-parameter tuning of an algorithm is a troublesome issue in black-box optimization. For CMA-ES, we needed to make a decision whether we use the plain CMA or the separable CMA, based on the limited knowledge on a problem of interest. If we select the separable CMA but the objective function is non-separable and highly ill-conditioned, we will not obtain a reasonable solution within most given function evaluation budgets. Therefore, plain CMA is a safer choice, though it may be slower on nearly separable functions. The dd-CMA automizes this decision and achieves faster adaptation speed on functions with ill-conditioned variables usually without compromising the performance on non-separable and highly ill-conditioned functions. Moreover, the advantage of dd-CMA is not limited to separable problems. We found a class of non-separable problems with variables of different sensitivity on which dd-CMA-ES decidedly outperforms CMA-ES *and* separable CMA-ES (see Figure 8a). As dd-CMA-ES improves the performance on a relatively wide range of problems with mis-scaling of variables, it is a prime candidate to become the default CMA-ES variant.

8 Acknowledgements

The authors thank the associate editor and the anonymous reviewers for their valuable comments and suggestions. This work is partly supported by JSPS KAKENHI Grant Number 19H04179.

References

Akimoto, Y., Auger, A., and Hansen, N. (2014). Comparison-based natural gradient optimization in high dimension. In *Proceedings of Genetic and Evolutionary Computation Conference*, pages 373–380.

Akimoto, Y., Auger, A., and Hansen, N. (2018). Quality gain analysis of the weighted recombination evolution strategy on general convex quadratic functions. *Theoretical Computer Science*.

Akimoto, Y. and Hansen, N. (2016). Projection-based restricted covariance matrix adaptation for high dimension. In *Genetic and Evolutionary Computation Conference, GECCO 2016, Denver, Colorado, USA, July 20-24, 2016*, pages 197–204. ACM.

Akimoto, Y., Nagata, Y., Ono, I., and Kobayashi, S. (2010). Bidirectional relation between CMA evolution strategies and natural evolution strategies. In *Parallel Problem Solving from Nature - PPSN XI*, number 6238 in LNCS, pages 154–163. Springer-Verlag.

Akimoto, Y., Sakuma, J., Ono, I., and Kobayashi, S. (2008). Functionally specialized CMA-ES: a modification of CMA-ES based on the specialization of the functions of covariance matrix adaptation and step size adaptation. In *GECCO '08: Proceedings of the 10th annual conference on Genetic and evolutionary computation*. ACM.

Arnold, D. V. (2006). Weighted multirecombination evolution strategies. *Theoretical Computer Science*, 361:18–37.

Arnold, D. V. and Hansen, N. (2010). Active covariance matrix adaptation for the (1+1)-CMA-ES. In *Proceedings of Genetic and Evolutionary Computation Conference*, pages 385–392, Portland, Oregon. ACM.

Auger, A. and Hansen, N. (2005a). Performance evaluation of an advanced local search evolutionary algorithm. In *2005 IEEE Congress on Evolutionary Computation*, pages 1777–1784. Ieee.

Auger, A. and Hansen, N. (2005b). A restart CMA evolution strategy with increasing population size. In *2005 IEEE Congress on Evolutionary Computation*, pages 1769–1776. Ieee.

Beyer, H.-G. (1995). Toward a theory of evolution strategies: On the benefits of sex—the $(\mu/\mu, \lambda)$ theory. *Evolutionary Computation*, 3(1):81–111.

Glasmachers, T., Schaul, T., Yi, S., Wierstra, D., and Schmidhuber, J. (2010). Exponential natural evolution strategies. In *Proceedings of Genetic and Evolutionary Computation Conference*, pages 393–400. ACM.

Hansen, N. (2000). Invariance, self-adaptation and correlated mutations in evolution strategies. In Schoenauer, M., Deb, K., Rudolph, G., Yao, X., Lutton, E., Guerv o s, J. J. M., and Schwefel, H.-P., editors, *Parallel Problem Solving from Nature - PPSN VI*, pages 355–364. Springer.

Hansen, N. (2009). Benchmarking a bi-population CMA-ES on the bbob-2009 function testbed. In *Workshop Proceedings of the GECCO Genetic and Evolutionary Computation Conference*, pages 2389–2395, New York, New York, USA. ACM Press.

Hansen, N., Arnold, D. V., and Auger, A. (2015). Evolution strategies. In Kacprzyk, J. and Pedrycz, W., editors, *Handbook of Computational Intelligence*, pages 871–898. Springer.

Hansen, N. and Auger, A. (2014). Principled design of continuous stochastic search: From theory to practice. In Borenstein, Y. and Moraglio, A., editors, *Theory and Principled Methods for the Design of Metaheuristics*. Springer.

Hansen, N. and Kern, S. (2004). Evaluating the CMA evolution strategy on multimodal test functions. In *Parallel Problem Solving from Nature - PPSN VIII*, pages 282–291. Springer.

Hansen, N., Muller, S. D., and Koumoutsakos, P. (2003). Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES). *Evolutionary Computation*, 11(1):1–18.

Hansen, N. and Ostermeier, A. (2001). Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation*, 9(2):159–195.

Hansen, N., Ros, R., Mauny, N., Schoenauer, M., and Auger, A. (2011). Impacts of invariance in search: When CMA-ES and PSO face ill-conditioned and non-separable problems. *Applied Soft Computing*, 11(8):5755–5769.

Harik, G. R. and Lobo, F. G. (1999). A parameter-less genetic algorithm. In *Proceedings of the 1st Annual Conference on Genetic and Evolutionary Computation - Volume 1*, GECCO'99, pages 258–265, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Harville, D. A. (2008). *Matrix Algebra from a Statistician's Perspective*. Springer-Verlag.

Heijmans, R. (1999). When does the expectation of a ratio equal the ratio of expectations? *Statistical Papers*, 40:107–115.

Jastrebski, G. and Arnold, D. V. (2006). Improving evolution strategies through active covariance matrix adaptation. In *2006 IEEE Congress on Evolutionary Computation*, pages 9719–9726. IEEE.

Karafotias, G., Hoogendoorn, M., and Eiben, A. E. (2015). Parameter control in evolutionary algorithms: Trends and challenges. *IEEE Transactions on Evolutionary Computation*, 19(2):167–187.

Knight, J. N. and Lunacek, M. (2007). Reducing the space-time complexity of the cma-es. In *Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation*, GECCO '07, pages 658–665, New York, NY, USA. ACM.

Krause, O., Arbonès, D. R., and Igel, C. (2016). Cma-es with optimal covariance update and storage complexity. In *Advances in Neural Information Processing Systems*, pages 370–378.

Krause, O. and Glasmachers, T. (2015). A CMA-ES with multiplicative covariance matrix updates. In *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*, GECCO '15, pages 281–288, New York, NY, USA. ACM.

Loshchilov, I. (2017). LM-CMA: An alternative to L-BFGS for large-scale black box optimization. *Evol. Comput.*, 25(1):143–171.

Loshchilov, I., Schoenauer, M., and Sebag, M. (2012). Alternative restart strategies for cma-es. In *International Conference on Parallel Problem Solving from Nature*, pages 296–305. Springer.

Ollivier, Y., Arnold, L., Auger, A., and Hansen, N. (2017). Information-geometric optimization algorithms: A unifying picture via invariance principles. *Journal of Machine Learning Research*, 18(18):1–65.

Ostermeier, A., Gawelczyk, A., and Hansen, N. (1994). A derandomized approach to self-adaptation of evolution strategies. *Evol. Comput.*, 2(4):369–380.

Price, K. V. (1997). Differential evolution vs. the functions of the 2nd ICEO. In *Evolutionary Computation, 1997., IEEE International Conference on*, pages 153–157. IEEE.

Rios, L. M. and Sahinidis, N. V. (2013). Derivative-free optimization: A review of algorithms and comparison of software implementations. *Journal of Global Optimization*, 56(3):1247–1293.

Ros, R. (2009). Benchmarking sep-CMA-ES on the BBOB-2009 function testbed. In *Proceedings of the 11th Annual Conference Companion on Genetic and Evolutionary Computation Conference: Late Breaking Papers*, GECCO '09, pages 2435–2440, New York, NY, USA. ACM.

Ros, R. and Hansen, N. (2008). A simple modification in CMA-ES achieving linear time and space complexity. In *Parallel Problem Solving from Nature - PPSN X*, pages 296–305. Springer.

Sun, Y., Wierstra, D., Schaul, T., and Schmidhuber, J. (2009). Efficient natural evolution strategies. In *Proceedings of the 11th Annual conference on Genetic and evolutionary computation - GECCO '09*, pages 539–545, New York, New York, USA. ACM Press.

Wierstra, D., Schaul, T., Glasmachers, T., Sun, Y., Peters, J., and Schmidhuber, J. (2014). Natural evolution strategies. *Journal of Machine Learning Research*, 15:949–980.

Wierstra, D., Schaul, T., Peters, J., and Schmidhuber, J. (2008). Natural evolution strategies. In *IEEE Congress on Evolutionary Computation*, pages 3381–3387.