CMA-ES with Optimal Covariance Update and Storage Complexity

Oswin Krause
Dept. of Computer Science
University of Copenhagen
Copenhagen, Denmark
oswin.krause@di.ku.dk

Dídac R. Arbonès
Dept. of Computer Science
University of Copenhagen
Copenhagen, Denmark
didac@di.ku.dk

Christian Igel
Dept. of Computer Science
University of Copenhagen
Copenhagen, Denmark
igel@di.ku.dk

Abstract

The covariance matrix adaptation evolution strategy (CMA-ES) is arguably one of the most powerful real-valued derivative-free optimization algorithms, finding many applications in machine learning. The CMA-ES is a Monte Carlo method, sampling from a sequence of multi-variate Gaussian distributions. Given the function values at the sampled points, updating and storing the covariance matrix dominates the time and space complexity in each iteration of the algorithm. We propose a numerically stable quadratic-time covariance matrix update scheme with minimal memory requirements based on maintaining triangular Cholesky factors. This requires a modification of the cumulative step-size adaption (CSA) mechanism in the CMA-ES, in which we replace the inverse of the square root of the covariance matrix by the inverse of the triangular Cholesky factor. Because the triangular Cholesky factor changes smoothly with the matrix square root, this modification does not change the behavior of the CMA-ES in terms of required objective function evaluations as verified empirically. Thus, the described algorithm can and should replace the standard CMA-ES if updating and storing the covariance matrix matters.

1 Introduction

The *covariance matrix adaptation evolution strategy*, CMA-ES [Hansen and Ostermeier, 2001], is recognized as one of the most competitive derivative-free algorithms for real-valued optimization [Beyer, 2007; Eiben and Smith, 2015]. The algorithm has been successfully applied in many unbiased performance comparisons and numerous real-world applications. In machine learning, it is mainly used for direct policy search in reinforcement learning and hyperparameter tuning in supervised learning (e.g., see Gomez *et al.* [2008]; Heidrich-Meisner and Igel [2009a,b]; Igel [2010], and references therein).

The CMA-ES is a Monte Carlo method for optimizing functions $f : \mathbb{R}^d \to \mathbb{R}$. The objective function f does not need to be continuous and can be multi-modal, constrained, and disturbed by noise. In each iteration, the CMA-ES samples from a d-dimensional multivariate normal distribution, the *search distribution*, and ranks the sampled points according to their objective function values. The mean and the covariance matrix of the search distribution are then adapted based on the ranked points. Given the ranking of the sampled points, the runtime of one CMA-ES iteration is $\omega(d^2)$ because the square root of the covariance matrix is required, which is typically computed by an eigenvalue decomposition. If the objective function can be evaluated efficiently and/or d is large, the computation of the matrix square root can easily dominate the runtime of the optimization process.

Various strategies have been proposed to address this problem. The basic approach for reducing the runtime is to perform an update of the matrix only every $\tau \in \Omega(d)$ steps [Hansen and Ostermeier,

1996, 2001], effectively reducing the time complexity to $O(d^2)$. However, this forces the algorithm to use outdated matrices during most iterations and can increase the amount of function evaluations. Furthermore, it leads to an uneven distribution of computation time over the iterations. Another approach is to restrict the model complexity of the search distribution [Poland and Zell, 2001; Ros and Hansen, 2008; Sun *et al.*, 2013; Akimoto *et al.*, 2014; Loshchilov, 2014, 2015], for example, to consider only diagonal matrices [Ros and Hansen, 2008]. However, this can lead to a drastic increase in function evaluations needed to approximate the optimum if the objective function is not compatible with the restriction, for example, when optimizing highly non-separable problems while only adapting the diagonal of the covariance matrix [Omidvar and Li, 2011]. More recently, methods were proposed that update the Cholesky factor of the covariance matrix instead of the covariance matrix itself [Suttorp *et al.*, 2009; Krause and Igel, 2015]. This works well for some CMA-ES variations (e.g., the (1+1)-CMA-ES and the multi-objective MO-CMA-ES [Suttorp *et al.*, 2009; Krause and Igel, 2015; Bringmann *et al.*, 2013]), however, the original CMA-ES relies on the matrix square root, which cannot be replaced one-to-one by a Cholesky factor.

In the following, we explore the use of the triangular Cholesky factorization instead of the square root in the standard CMA-ES. In contrast to previous attempts in this direction, we present an approach that comes with a theoretical justification for why it does not deteriorate the algorithm's performance. This approach leads to the optimal asymptotic storage and runtime complexity when adaptation of the full covariance matrix is required, as is the case for non-separable ill-conditioned problems. Our CMA-ES variant, referred to as Cholesky-CMA-ES, reduces the runtime complexity of the algorithm with no significant change in the number of objective function evaluations. It also reduces the memory footprint of the algorithm.

Section 2 briefly describes the original CMA-ES algorithm (for details we refer Hansen [2015]). In section 3 we propose our new method for approximating the step-size adaptation. We give a theoretical justification for the convergence of the new algorithm. We provide empirical performance results comparing the original CMA-ES with the new Cholesky-CMA-ES using various benchmark functions in section 4. Finally, we discuss our results and draw our conclusions.

2 Background

Before we briefly describe the CMA-ES to fix our notation, we discuss some basic properties of using a Cholesky decomposition to sample from a multi-variate Gaussian distribution. Sampling from a d-dimensional multi-variate normal distribution $\mathcal{N}(\boldsymbol{m}, \Sigma)$, $\boldsymbol{m} \in \mathbb{R}^d$, $\Sigma \in \mathbb{R}^{d \times d}$ is usually done using a decomposition of the covariance matrix Σ. This could be the square root of the matrix $\Sigma = HH \in \mathbb{R}^{d \times d}$ or a lower triangular Cholesky factorization $\Sigma = AA^T$, which is related to the square root by the QR-decomposition $H = AE$ where E is an orthogonal matrix. We can sample a point $\boldsymbol{x}$ from $\mathcal{N}(\boldsymbol{m}, \Sigma)$ using a sample $\boldsymbol{z} \sim \mathcal{N}(\boldsymbol{0}, \mathrm{I})$ by $\boldsymbol{x} = H\boldsymbol{z} + \boldsymbol{m} = AE\boldsymbol{z} + \boldsymbol{m} = A\boldsymbol{y} + \boldsymbol{m}$, where we set $\boldsymbol{y} = E\boldsymbol{z}$. We have $\boldsymbol{y} \sim \mathcal{N}(\boldsymbol{0}, \mathrm{I})$ since E is orthogonal. Thus, as long as we are only interested in the value of $\boldsymbol{x}$ and do not need $\boldsymbol{y}$, we can sample using the Cholesky factor instead of the matrix square root.

2.1 CMA-ES

The CMA-ES has been proposed by Hansen and Ostermeier [1996, 2001] and its most recent version is described by Hansen [2015]. In the tth iteration of the algorithm, the CMA-ES samples λ points from a multivariate normal distribution $\mathcal{N}(\boldsymbol{m}_t, \sigma_t^2 \cdot C_t)$, evaluates the objective function f at these points, and adapts the parameters $C_t \in \mathbb{R}^{d \times d}$, $\boldsymbol{m}_t \in \mathbb{R}^d$, and $\sigma_t \in \mathbb{R}^+$. In the following, we present the update procedure in a slightly simplified form (for didactic reasons, we refer to Hansen [2015] for the details). All parameters (μ, λ, $\boldsymbol{\omega}$, c_σ, d_σ, c_c, c_1, c_μ) are set to their default values [Hansen, 2015, Table 1].

For a minimization task, the λ points are ranked by function value such that $f(\boldsymbol{x}_{1,t}) \leq f(\boldsymbol{x}_{2,t}) \leq \cdots \leq f(\boldsymbol{x}_{\lambda,t})$. The distribution mean is set to the weighted average $\boldsymbol{m}_{t+1} = \sum_{i=1}^{\mu} \omega_i \boldsymbol{x}_{i,t}$. The weights depend only on the ranking, not on the function values directly. This renders the algorithm invariant under order-preserving transformation of the objective function. Points with smaller ranks (i.e., better objective function values) are given a larger weight ω_i with $\sum_{i=1}^{\lambda} \omega_i = 1$. The weights are zero for ranks larger than $\mu < \lambda$, which is typically $\mu = \lambda/2$. Thus, points with function values worse than the median do not enter the adaptation process of the parameters. The covariance matrix

is updated using two terms, a rank-1 and a rank-μ update. For the rank-1 update, a long term average of the changes of m_t is maintained

$$p_{c,t+1} = (1 - c_c)p_{c,t} + \sqrt{c_c(2 - c_c)\mu_{\text{eff}}}\frac{m_{t+1} - m_t}{\sigma_t} \quad , \tag{1}$$

where $\mu_{\text{eff}} = 1/\sum_{i=1}^{\mu}\omega_i^2$ is the effective sample size given the weights. Note that $p_{c,t}$ is large when the algorithm performs steps in the same direction, while it becomes small when the algorithm performs steps in alternating directions.[1] The rank-μ update estimates the covariance of the weighted steps $x_{i,t} - m_t$, $1 \leq i \leq \mu$. Combining rank-1 and rank-μ update gives the final update rule for C_t, which can be motivated by principles from information geometry [Akimoto *et al.*, 2012]:

$$C_{t+1} = (1 - c_1 - c_\mu)C_t + c_1 p_{c,t+1}p_{c,t+1}^T + \frac{c_\mu}{\sigma_t^2}\sum_{i=1}^{\mu}\omega_i\left(x_{i,t} - m_t\right)\left(x_{i,t} - m_t\right)^T \tag{2}$$

So far, the update is (apart from initialization) invariant under affine linear transformations (i.e., $x \mapsto Bx + b$, $B \in \text{GL}(d, \mathbb{R})$).

The update of the global step-size parameter σ_t is based on the *cumulative step-size adaptation* algorithm (CSA). It measures the correlation of successive steps in a normalized coordinate system. The goal is to adapt σ_t such that the steps of the algorithm become uncorrelated. Under the assumption that uncorrelated steps are standard normally distributed, a carefully designed long term average over the steps should have the same expected length as a χ-distributed random variable, denoted by $\mathbb{E}\{\chi\}$. The long term average has the form

$$p_{\sigma,t+1} = (1 - c_\sigma)p_{\sigma,t} + \sqrt{c_\sigma(2 - c_\sigma)\mu_{\text{eff}}}\,C_t^{-1/2}\frac{m_{t+1} - m_t}{\sigma_t} \tag{3}$$

with $p_{\sigma,1} = 0$. The normalization by the factor $C_t^{-1/2}$ is the main difference between equations (1) and (3). It is important because it corrects for a change of C_t between iterations. Without this correction, it is difficult to measure correlations accurately in the un-normalized coordinate system. For the update, the length of $p_{\sigma,t+1}$ is compared to the expected length $\mathbb{E}\{\chi\}$ and σ_t is changed depending on whether the average step taken is longer or shorter than expected:

$$\sigma_{t+1} = \sigma_t \exp\left(\frac{c_\sigma}{d_\sigma}\left(\frac{\|p_{\sigma,t+1}\|}{\mathbb{E}\{\chi\}} - 1\right)\right) \tag{4}$$

This update is *not* proven to preserve invariance under affine linear transformations [Auger, 2015], and it is it conjectured that it does not.

3 Cholesky-CMA-ES

In general, computing the matrix square root or the Cholesky factor from an $n \times n$ matrix has time complexity $\omega(d^2)$ (i.e., scales worse than quadratically). To reduce this complexity, Suttorp *et al.* [2009] have suggested to replace the process of updating the covariance matrix and decomposing it afterwards by updates directly operating on the decomposition (i.e., the covariance matrix is never computed and stored explicitly, only its factorization is maintained). Krause and Igel [2015] have shown that the update of C_t in equation (2) can be rewritten as a quadratic-time update of its triangular Cholesky factor A_t with $C_t = A_t A_t^T$. They consider the special case $\mu = \lambda = 1$. We propose to extend this update to the standard CMA-ES, which leads to a runtime $O(\mu d^2)$. As typically $\mu = O(\log(d))$, this gives a large speed-up compared to the explicit recomputation of the Cholesky factor or the inverse of the covariance matrix.

Unfortunately, the fast Cholesky update can not be applied directly to the original CMA-ES. To see this, consider the term $s_t = C_t^{-1/2}(m_{t+1} - m_t)$ in equation (3). Rewriting $p_{\sigma,t+1}$ in terms of s_t in a non-recursive fashion, we obtain

$$p_{\sigma,t+1} = \sqrt{c_\sigma(2 - c_\sigma)\mu_{\text{eff}}}\sum_{k=1}^{t}\frac{(1 - c_\sigma)^{t-k}}{\sigma_k}s_k \quad .$$

---

[1]Given c_c, the factors in (1) are chosen to compensate for the change in variance when adding distributions. If the ranking of the points would be purely random, $\sqrt{\mu_{\text{eff}}} \cdot (m_{t+1} - m_t)/\sigma_t \sim \mathcal{N}(0, C_t)$ and if $C_t = I$ and $p_{c,t} \sim \mathcal{N}(0, \mathrm{I})$ then $p_{c,t+1} \sim \mathcal{N}(0, \mathrm{I})$.

Algorithm 1: The Cholesky-CMA-ES.

---

input : $\lambda,\,\mu,\,\boldsymbol{m}_1,\,\omega_{i=1\ldots\mu},\,c_\sigma,\,d_\sigma,\,c_c,\,c_1$ and c_μ

$A_1 = \mathrm{I},\, p_{c,1} = \boldsymbol{0},\, p_{\sigma,1} = \boldsymbol{0}$

for $t = 1, 2, \ldots$ **do**

    for $i = 1, \ldots, \lambda$ **do**

        $\boldsymbol{x}_{i,t} = \sigma_t A_t \boldsymbol{y}_{i,t} + \boldsymbol{m}_t,\ \boldsymbol{y}_{i,t} \sim \mathcal{N}(\boldsymbol{0}, \mathrm{I})$

    Sort $\boldsymbol{x}_{i,t}, i = 1, \ldots, \lambda$ increasing by $f(\boldsymbol{x}_{i,t})$

    $\boldsymbol{m}_{t+1} = \sum_{i=1}^{\mu} \omega_i \boldsymbol{x}_{i,t}$

    $\boldsymbol{p}_{c,t+1} = (1 - c_c)\boldsymbol{p}_{c,t} + \sqrt{c_c(2 - c_c)\mu_{\mathrm{eff}}}\,\frac{\boldsymbol{m}_{t+1} - \boldsymbol{m}_t}{\sigma_t}$

    // Apply formula (2) to A_t

    $A_{t+1} \leftarrow \sqrt{1 - c_1 - c_\mu}\,A_t$

    $A_{t+1} \leftarrow \mathrm{rankOneUpdate}(A_{t+1}, c_1, p_{c,t+1})$

    for $i = 1, \ldots, \mu$ **do**

        $A_{t+1} \leftarrow \mathrm{rankOneUpdate}(A_{t+1}, c_\mu \omega_i, \frac{\boldsymbol{x}_{i,t} - \boldsymbol{m}_t}{\sigma_t})$

    // Update σ using $\hat{s}_k$ as in (5)

    $\boldsymbol{p}_{\sigma,t+1} = (1 - c_\sigma)\boldsymbol{p}_{\sigma,t} + \sqrt{c_\sigma(2 - c_\sigma)\mu_{\mathrm{eff}}}\,A_t^{-1}\frac{\boldsymbol{m}_{t+1} - \boldsymbol{m}_t}{\sigma_t}$

    $\sigma_{t+1} = \sigma_t \exp\left(\frac{c_\sigma}{d_\sigma}\left(\frac{\|\boldsymbol{p}_{\sigma,t+1}\|}{\mathbb{E}\{\chi\}} - 1\right)\right)$

Algorithm 2: rankOneUpdate$(A, \beta, \boldsymbol{v})$

---

input : Cholesky factor $A \in \mathbb{R}^{d \times d}$ of C, $\beta \in \mathbb{R}$, $\boldsymbol{v} \in \mathbb{R}^d$

output : Cholesky factor A' of $C + \beta \boldsymbol{v}\boldsymbol{v}^T$

$\boldsymbol{\alpha} \leftarrow \boldsymbol{v}$

$b \leftarrow 1$

for $j = 1, \ldots, d$ **do**

    $A'_{jj} \leftarrow \sqrt{A_{jj}^2 + \frac{\beta}{b}\alpha_j^2}$

    $\gamma \leftarrow A_{jj}^2 b + \beta \alpha_j^2$

    for $k = j+1, \ldots, d$ **do**

        $\alpha_k \leftarrow \alpha_k - \frac{\alpha_j}{A_{jj}}A_{kj}$

        $A'_{kj} = \frac{A'_{jj}}{A_{jj}}A_{kj} + \frac{A'_{jj}\beta\alpha_j}{\gamma}\alpha_k$

    $b \leftarrow b + \beta\frac{\alpha_j^2}{A_{jj}^2}$

By the RQ-decomposition, we can find $C_t^{1/2} = A_t E_t$ with E_t being an orthogonal matrix and A_t lower triangular. When replacing $\boldsymbol{s}_t$ by $\hat{\boldsymbol{s}}_t = A_t^{-1}(\boldsymbol{m}_{t+1} - \boldsymbol{m}_t)$, we obtain

$$\boldsymbol{p}_{\sigma,t+1} = \sqrt{c_\sigma(2 - c_\sigma)\mu_{\mathrm{eff}}} \sum_{k=1}^{t} \frac{(1 - c_\sigma)^{t-k}}{\sigma_k} E_k^T \hat{\boldsymbol{s}}_k \ .$$

Thus, replacing $C_t^{-1/2}$ by A_t^{-1} introduces a new random rotation matrix E_t^T, which changes in every iteration. Obtaining E_t from A_t can be achieved by the polar-decomposition, which is a cubic-time operation: currently there are no algorithms known that can update an existing polar decomposition from an updated Cholesky factor in less than cubic time. Thus, if our goal is to apply the fast Cholesky update, we have to perform the update without this correction factor

$$\boldsymbol{p}_{\sigma,t+1} \approx \sqrt{c_\sigma(2 - c_\sigma)\mu_{\mathrm{eff}}} \sum_{k=1}^{t} \frac{(1 - c_\sigma)^{t-k}}{\sigma_k} \hat{\boldsymbol{s}}_k \ . \tag{5}$$

This introduces some error, but we will show in the following that we can expect this error to be small and to decrease over time as the algorithm converges to the optimum. For this, we need the following result:

4

Lemma 1. *Consider the sequence of symmetric positive definite matrices $\bar{C}_{t=0}^{\infty}$ with $\bar{C}_t = C_t (\det C_t)^{-1/d}$. Assume that $\bar{C}_t \overset{t \to \infty}{\Longrightarrow} \bar{C}$ and that $\bar{C}$ is symmetric positive definite with $\det \bar{C} = 1$. Let $\bar{C}_t^{1/2} = \bar{A}_t E_t$ denote the RQ-decomposition of $\bar{C}_t^{1/2}$, where E_t is orthogonal and $\bar{A}_t$ lower triangular. Then it holds $E_{t-1}^T E_t \overset{t \to \infty}{\Longrightarrow} I$.*

Proof. Let $\bar{C} = \bar{A}E$, the RQ-decomposition of $\bar{C}$. As $\det \bar{C} \neq 0$, this decomposition is unique. Because the RQ-decomposition is continuous, it maps convergent sequences to convergent sequences. Therefore $E_t \overset{t \to \infty}{\Longrightarrow} E$ and thus, $E_{t-1}^T E_t \overset{t \to \infty}{\Longrightarrow} E^T E = I$. $\qquad\square$

This result establishes that, when C_t converges to a certain shape (but not necessary to a certain scaling), A_t and thus E_t will also converge (up to scaling). Thus, as we only need the norm of $\boldsymbol{p}_{\sigma,t+1}$, we can rotate the coordinate system and by multiplying with E_t we obtain

$$\|\boldsymbol{p}_{\sigma,t+1}\| = \|E_t \boldsymbol{p}_{\sigma,t+1}\| = \sqrt{c_\sigma (2 - c_\sigma)\mu_{\text{eff}}} \left\| \sum_{k=1}^{t} \frac{(1 - c_\sigma)^{t-k}}{\sigma_k} E_t E_k^T \hat{\boldsymbol{s}}_k \right\| . \tag{6}$$

Therefore, if $E_t E_{t-1}^T \overset{t \to \infty}{\Longrightarrow} I$, the error in the norm will also vanish due to the exponential weighting in the summation. Note that this does not hold for any decomposition $C_t = B_t B_t^T$. If we do not constrain B_t to be triangular and allow any matrix, we do not have a bijective mapping between C_t and B_t anymore and the introduction of $\frac{d(d-1)}{2}$ degrees of freedom (as, e.g., in the update proposed by Suttorp *et al.* [2009]) allows the creation of non-converging sequences of E_t even for $C_t = \text{const}$.

As the CMA-ES is a randomized algorithm, we cannot assume convergence of C_t. However, in simplified algorithms the expectation of C_t converges [Beyer, 2014]. Still, the reasoning behind Lemma 1 establishes that the error caused by replacing $\boldsymbol{s}_t$ by $\hat{\boldsymbol{s}}_t$ is small if C_t changes slowly. Equation (6) establishes that the error depends only on the rotation of coordinate systems. As the mapping from C_t to the triangular factor A_t is one-to-one and smooth, the coordinate system changes in every step will be small – and because of the exponentially decaying weighting, only the last few coordinate systems matter at a particular time step t.

The Cholesky-CMA-ES algorithm is given in Algorithm 1. One can derive the algorithm from the standard CMA-ES by decomposing (2) into a number of rank-1 updates $C_{t+1} = (((\alpha C_t + \beta_1 \boldsymbol{v}_1 \boldsymbol{v}_1^T) + \beta_2 \boldsymbol{v}_2 \boldsymbol{v}_2^T) \dots)$ and applying them to the Cholesky factor using Algorithm 2.

Properties of the update rule. The $O(\mu d^2)$ complexity of the update in the Cholesky-CMA-ES is asymptotically optimal.[2] Apart from the theoretical guarantees, there are several additional advantages compared to approaches using a non-triangular Cholesky factorization (e.g., Suttorp *et al.* [2009]). First, as only triangular matrices have to be stored, the storage complexity is optimal. Second, the diagonal elements of a triangular Cholesky factor are the square roots of the eigenvalues of the factorized matrix, that is, we get the eigenvalues of the covariance matrix for free. These are important, for example, for monitoring the conditioning of the optimization problem and, in particular, to enforce lower bounds on the variances of $\sigma_t C_t$ projected on its principal components. Third, a triangular matrix can be inverted in quadratic time. Thus, we can efficiently compute A_t^{-1} from A_t when needed, instead of having two separate quadratic-time updates for A_t^{-1} and A_t, which requires more memory and is prone to numerical instabilities.

4 Experiments and Results

Experiments. We compared the Cholesky-CMA-ES with other CMA-ES variants.[3] The reference CMA-ES implementation uses a delay strategy in which the matrix square root is computed every $\max \left\{ 1, \frac{1}{10d(c_1 + c_\mu)} \right\}$ iterations [Hansen, 2015], which equals one for the dimensions considered

---

[2]Actually, the complexity is related to the complexity of multiplying two $\mu \times d$ matrices. We assume a naïve implementation of matrix multiplication. With a faster multiplication algorithm, the complexity can be reduced accordingly.

[3]We added our algorithm to the open-source machine learning library Shark [Igel *et al.*, 2008] and used LAPACK for high efficiency.
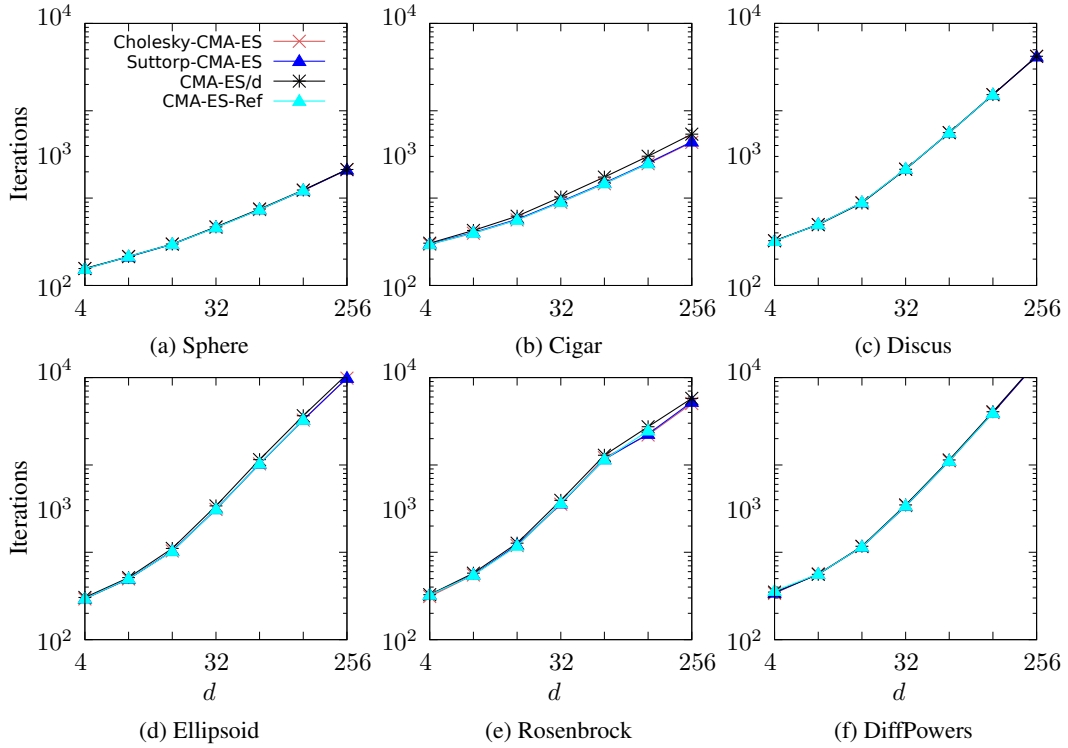
Figure 1: Function evaluations required to reach $f(\boldsymbol{x}) < 10^{-14}$ over problem dimensionality (medians of 100 trials). The graphs for CMA-ES-Ref and Cholesky-CMA-ES overlap.
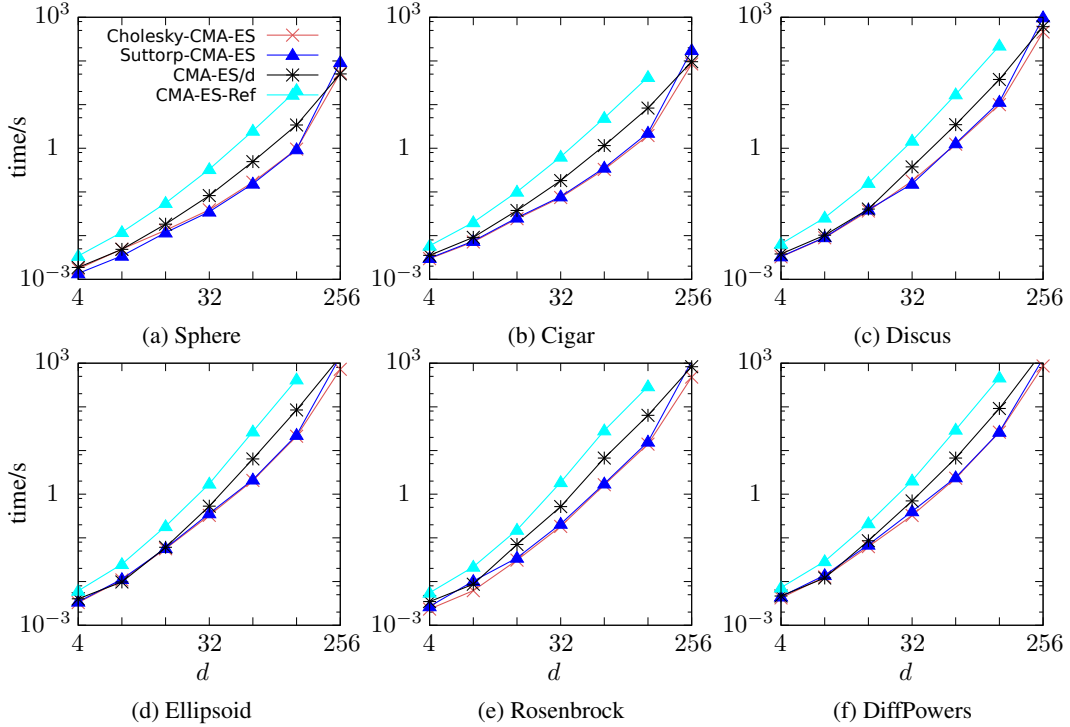

Figure 2: Runtime in seconds over problem dimensionality. Shown are medians of 100 trials. Note the logarithmic scaling on both axes.

Name	$f(\boldsymbol{x})$
Sphere	$\|\boldsymbol{x}\|^2$
Rosenbrock	$\sum_{i=0}^{d-1}\left(100(x_{i+1}-x_i^2)^2+(1-x_i)^2\right)$
Discus	$x_0^2+\sum_{i=1}^{d}10^{-6}x_i^2$
Cigar	$10^{-6}x_0^2+\sum_{i=1}^{d}x_i^2$
Ellipsoid	$\sum_{i=0}^{d}10^{\frac{-6i}{d-1}}x_i^2$
Different Powers	$\sum_{i=0}^{d}|x_i|^{\frac{2+10i}{d-1}}$

Table 1: Benchmark functions used in the experiments (additionally, a rotation matrix B transforms the variables, $\boldsymbol{x} \mapsto B\boldsymbol{x}$)
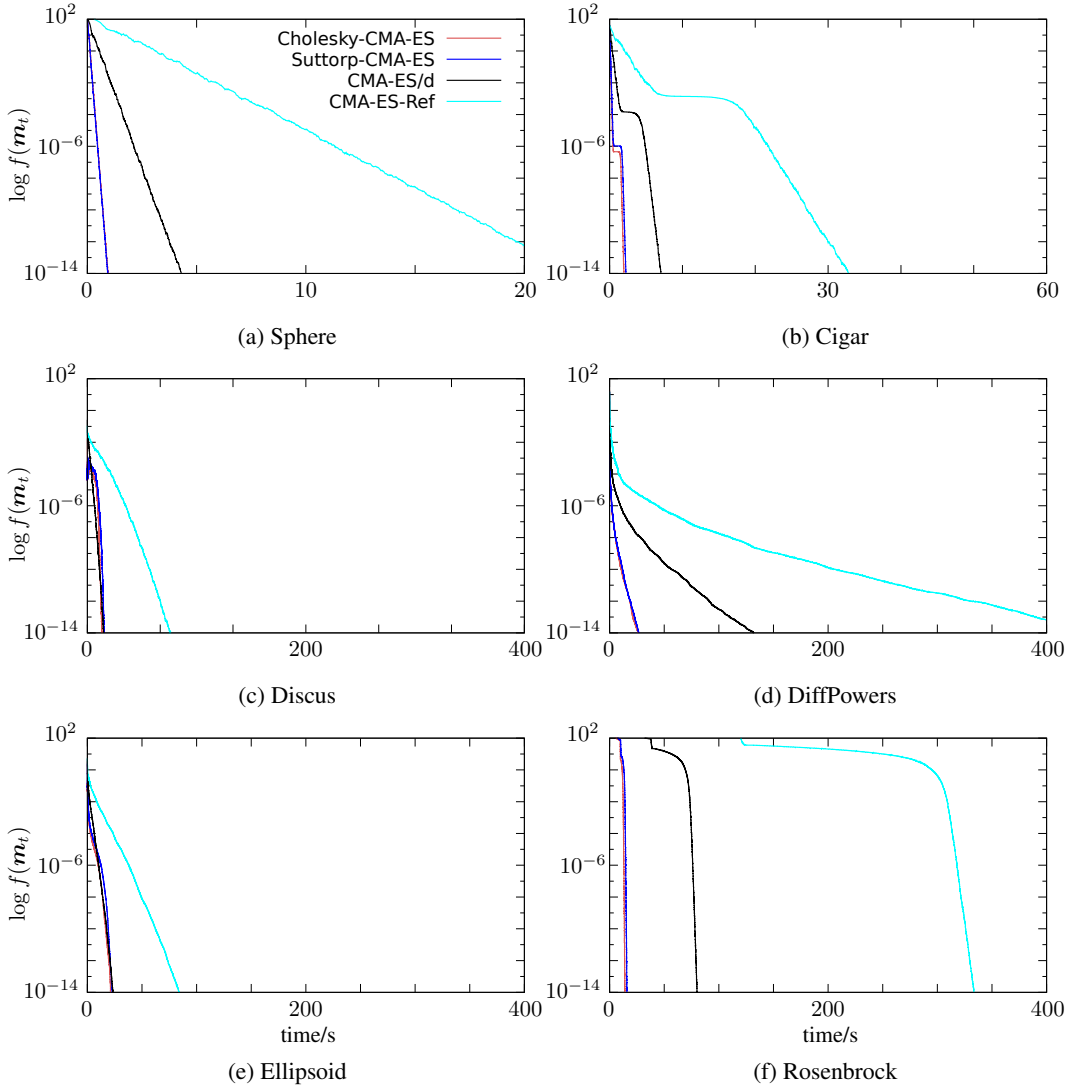

Figure 3: Function value evolution over time on the benchmark functions with $d = 128$. Shown are single runs, namely those with runtimes closest to the corresponding median runtimes.

7

in our experiments. We call this variant CMA-ES-Ref. As an alternative, we experimented with delaying the update for d steps. We refer to this variant as CMA-ES/d. We also adapted the non-triangular Cholesky factor approach by Suttorp *et al.* [2009] to the state-of-the art implementation of the CMA-ES. We refer to the resulting algorithm as Suttorp-CMA-ES.

We considered standard benchmark functions for derivative-free optimization given in Table 1. Sphere is considered to show that on a spherical function the step size adaption does not behave differently; Cigar/Discus/Ellipsoid model functions with different convex shapes near the optimum; Rosenbrock tests learning a function with $d - 1$ bends, which lead to slowly converging covariance matrices in the optimization process; Diffpowers is an example of a function with arbitrarily bad conditioning.

To test rotation invariance, we applied a rotation matrix to the variables, $\boldsymbol{x} \mapsto B\boldsymbol{x}$, $B \in \mathrm{SO}(d, \mathbb{R})$. This is done for every benchmark function, and a rotation matrix was chosen randomly at the beginning of each trial. All starting points were drawn uniformly from $[0, 1]$, except for Sphere, where we sampled from $\mathcal{N}(0, \mathrm{I})$. For each function, we vary $d \in \{4, 8, 16, \ldots, 256\}$. Due to the long running times, we only compute CMA-ES-Ref up to $d = 128$. For the given range of dimensions, for every choice of d, we ran 100 trials from different initial points and monitored the number of iterations and the wall-clock time needed to sample a point with a function value below 10^{-14}. For Rosenbrock we excluded the trials in which the algorithm did not converge to the global optimum.

We further evaluated the algorithms on additional benchmark functions inspired by Stich and Müller [2012] and measured the change of rotation introduced by the Cholesky-CMA-ES at each iteration (E_t), see supplementary material.

Results. Figure 1 shows that CMA-ES-Ref and Cholesky-CMA-ES required the same amount of function evaluations to reach a given objective value. The CMA-ES/d required slightly more evaluations depending on the benchmark function. When considering the wall-clock runtime, the Cholesky-CMA-ES was significantly faster than the other algorithms. As expected from the theoretical analysis, the higher the dimensionality the more pronounced the differences, see Figure 2 (note logarithmic scales). For $d = 64$ the Cholesky-CMA-ES was already 20 times faster than the CMA-ES-Ref. The drastic differences in runtime become apparent when inspecting single trials. Note that for $d = 256$ the matrix size exceeded the L2 cache, which affected the performance of the Cholesky-CMA-ES and Suttorp-CMA-ES. Figure 3 plots the trials with runtimes closest to the corresponding median runtimes for $d = 128$.

5 Conclusion

CMA-ES is a ubiquitous algorithm for derivative-free optimization. The CMA-ES has proven to be a highly efficient direct policy search algorithm and to be a useful tool for model selection in supervised learning. We propose the Cholesky-CMA-ES, which can be regarded as an approximation of the original CMA-ES. We gave theoretical arguments for why our approximation, which only affects the global step-size adaptation, does not impair performance. The Cholesky-CMA-ES achieves a better, asymptotically optimal time complexity of $O(\mu d^2)$ for the covariance update and optimal memory complexity. It allows for numerically stable computation of the inverse of the Cholesky factor in quadratic time and provides the eigenvalues of the covariance matrix without additional costs. We empirically compared the Cholesky-CMA-ES to the state-of-the-art CMA-ES with delayed covariance matrix decomposition. Our experiments demonstrated a significant increase in optimizaton speed. As expected, the Cholesky-CMA-ES needed the same amount of objective function evaluations as the standard CMA-ES, but required much less wall-clock time – and this speed-up increases with the search space dimensionality. Still, our algorithm scales quadratically with the problem dimensionality. If the dimensionality gets so large that maintaining a full covariance matrix becomes computationally infeasible, one has to resort to low-dimensional approximations [e.g., Loshchilov, 2015], which, however, bear the risk of a significant drop in optimization performance. Thus, we advocate our new Cholesky-CMA-ES for scaling up CMA-ES to large optimization problems for which updating and storing the covariance matrix is still possible, for example, for training neural networks in direct policy search.

References

Y. Akimoto, Y. Nagata, I. Ono, and S. Kobayashi. Theoretical foundation for CMA-ES from information geometry perspective. *Algorithmica*, 64(4):698–716, 2012.

Y. Akimoto, A. Auger, and N. Hansen. Comparison-based natural gradient optimization in high dimension. In *Proceedings of the 16th Annual Genetic and Evolutionary Computation Conference (GECCO)*, pages 373–380. ACM, 2014.

A. Auger. *Analysis of Comparison-based Stochastic Continous Black-Box Optimization Algorithms*. Habilitation thesis, Faculté des Sciences d'Orsay, Université Paris-Sud, 2015.

H.-G. Beyer. Evolution strategies. *Scholarpedia*, 2(8):1965, 2007.

H.-G. Beyer. Convergence analysis of evolutionary algorithms that are based on the paradigm of information geometry. *Evolutionary Computation*, 22(4):679–709, 2014.

K. Bringmann, T. Friedrich, C. Igel, and T. Voß. Speeding up many-objective optimization by Monte Carlo approximations. *Artificial Intelligence*, 204:22–29, 2013.

A. E. Eiben and Jim Smith. From evolutionary computation to the evolution of things. *Nature*, 521:476–482, 2015.

F. Gomez, J. Schmidhuber, and R. Miikkulainen. Accelerated neural evolution through cooperatively coevolved synapses. *Journal of Machine Learning Research*, 9:937–965, 2008.

N. Hansen and A. Ostermeier. Adapting arbitrary normal mutation distributions in evolution strategies: The covariance matrix adaptation. In *Proceedings of IEEE International Conference on Evolutionary Computation (CEC 1996)*, pages 312–317. IEEE, 1996.

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

N. Hansen. The CMA evolution strategy: A tutorial. Technical report, Inria Saclay – Île-de-France, Université Paris-Sud, LRI, 2015.

V. Heidrich-Meisner and C. Igel. Hoeffding and Bernstein races for selecting policies in evolutionary direct policy search. In *Proceedings of the 26th International Conference on Machine Learning (ICML 2009)*, pages 401–408, 2009.

V. Heidrich-Meisner and C. Igel. Neuroevolution strategies for episodic reinforcement learning. *Journal of Algorithms*, 64(4):152–168, 2009.

C. Igel, T. Glasmachers, and V. Heidrich-Meisner. Shark. *Journal of Machine Learning Research*, 9:993–996, 2008.

C. Igel. Evolutionary kernel learning. In *Encyclopedia of Machine Learning*. Springer-Verlag, 2010.

O. Krause and C. Igel. A more efficient rank-one covariance matrix update for evolution strategies. In *Proceedings of the 2015 ACM Conference on Foundations of Genetic Algorithms (FOGA XIII)*, pages 129–136. ACM, 2015.

I. Loshchilov. A computationally efficient limited memory CMA-ES for large scale optimization. In *Proceedings of the 16th Annual Genetic and Evolutionary Computation Conference (GECCO)*, pages 397–404. ACM, 2014.

I. Loshchilov. LM-CMA: An alternative to L-BFGS for large scale black-box optimization. *Evolutionary Computation*, 2015.

M. N. Omidvar and X. Li. A comparative study of CMA-ES on large scale global optimisation. In *AI 2010: Advances in Artificial Intelligence*, volume 6464 of *LNAI*, pages 303–312. Springer, 2011.

J. Poland and A. Zell. Main vector adaptation: A CMA variant with linear time and space complexity. In *Proceedings of the 10th Annual Genetic and Evolutionary Computation Conference (GECCO)*, pages 1050–1055. Morgan Kaufmann Publishers, 2001.

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

S. U. Stich and C. L. Müller. On spectral invariance of randomized Hessian and covariance matrix adaptation schemes. In *Parallel Problem Solving from Nature (PPSN XII)*, pages 448–457. Springer, 2012.

Y. Sun, T. Schaul, F. Gomez, and J. Schmidhuber. A linear time natural evolution strategy for non-separable functions. In *15th Annual Conference on Genetic and Evolutionary Computation Conference Companion*, pages 61–62. ACM, 2013.

T. Suttorp, N. Hansen, and C. Igel. Efficient covariance matrix update for variable metric evolution strategies. *Machine Learning*, 75(2):167–197, 2009.