The CMA Evolution Strategy: A Tutorial

Nikolaus Hansen

June 28, 2011

Contents

Nomenclature **3**

0 Preliminaries **4**
0.1 Eigendecomposition of a Positive Definite Matrix 5
0.2 The Multivariate Normal Distribution 6
0.3 Randomized Black Box Optimization 7
0.4 Hessian and Covariance Matrices 8

1 Basic Equation: Sampling **8**

2 Selection and Recombination: Moving the Mean **9**

3 Adapting the Covariance Matrix **10**
3.1 Estimating the Covariance Matrix From Scratch 10
3.2 Rank-μ-Update 12
3.3 Rank-One-Update 14
3.3.1 A Different Viewpoint 14
3.3.2 Cumulation: Utilizing the Evolution Path 16
3.4 Combining Rank-μ-Update and Cumulation 17

4 Step-Size Control **17**

5 Discussion **21**

A Algorithm Summary: The CMA-ES **25**

B Implementational Concerns **27**
B.1 Multivariate normal distribution 28
B.2 Strategy internal numerical effort 28
B.3 Termination criteria 28
B.4 Flat fitness 29
B.5 Boundaries and Constraints. 29

C MATLAB Source Code **31**

D Reformulation of Learning Parameter c_{cov} **33**

Nomenclature

We adopt the usual vector notation, where bold letters, $\boldsymbol{v}$, are column vectors, capital bold letters, $\boldsymbol{A}$, are matrices, and a transpose is denoted by $\boldsymbol{v}^{\mathrm{T}}$. A list of used abbreviations and symbols is given in alphabetical order.

Abbreviations

CMA Covariance Matrix Adaptation

EMNA Estimation of Multivariate Normal Algorithm

ES Evolution Strategy

$(\mu/\mu_{\{\mathrm{I,W}\}}, \lambda)$-ES, Evolution Strategy with μ parents, with recombination of all μ parents, either Intermediate or Weighted, and λ offspring.

RHS Right Hand Side.

Greek symbols

$\lambda \geq 2$, population size, sample size, number of offspring, see (5).

$\mu \leq \lambda$ parent number, number of selected search points in the population, see (6).

$\mu_{\mathrm{eff}} = \left(\sum_{i=1}^{\mu} w_i^2\right)^{-1} = 1/\|\boldsymbol{w}\|^2$, the variance effective selection mass, see (8).

$\sigma^{(g)} \in \mathbb{R}_+$, step-size.

Latin symbols

$\boldsymbol{B} \in \mathbb{R}^n$, an orthogonal matrix. Columns of $\boldsymbol{B}$ are eigenvectors of $\boldsymbol{C}$ with unit length and correspond to the diagonal elements of $\boldsymbol{D}$.

$\boldsymbol{C}^{(g)} \in \mathbb{R}^{n\times n}$, covariance matrix at generation g.

c_{ii}, diagonal elements of $\boldsymbol{C}$.

$c_{\mathrm{c}} \leq 1$, learning rate for cumulation for the rank-one update of the covariance matrix, see (22) and (42), and Table 1.

$c_1 \leq 1 - c_\mu$, learning rate for the rank-one update of the covariance matrix update, see (26), (27), and (43), and Table 1.

$c_\mu \leq 1 - c_1$, learning rate for the rank-μ update of the covariance matrix update, see (14), (27), and (43), and Table 1.

$c_\sigma < 1$, learning rate for the cumulation for the step-size control, see (28) and (40), and Table 1.

$\boldsymbol{D} \in \mathbb{R}^n$, a diagonal matrix. The diagonal elements of $\boldsymbol{D}$ are square roots of eigenvalues of $\boldsymbol{C}$ and correspond to the respective columns of $\boldsymbol{B}$.

$d_i > 0$, diagonal elements of diagonal matrix $\boldsymbol{D}$, d_i^2 are eigenvalues of $\boldsymbol{C}$.

$d_\sigma \approx 1$, damping parameter for step-size update, see (29), (34), and (41).

E Expectation value

$f : \mathbb{R}^n \to \mathbb{R}, \boldsymbol{x} \mapsto f(\boldsymbol{x})$, objective function (fitness function) to be minimized.

$f_{\text{sphere}} : \mathbb{R}^n \to \mathbb{R}, \boldsymbol{x} \mapsto \|\boldsymbol{x}\|^2 = \boldsymbol{x}^{\mathrm{T}}\boldsymbol{x} = \sum_{i=1}^{n} x_i^2$.

$g \in \mathbb{N}_0$, generation counter, iteration number.

$\mathbf{I} \in \mathbb{R}^{n\times n}$, Identity matrix, unity matrix.

$\boldsymbol{m}^{(g)} \in \mathbb{R}^n$, mean value of the search distribution at generation g.

$n \in \mathbb{N}$, search space dimension, see f.

$\mathcal{N}(\mathbf{0}, \mathbf{I})$, multivariate normal distribution with zero mean and unity covariance matrix. A vector distributed according to $\mathcal{N}(\mathbf{0}, \mathbf{I})$ has independent, $(0, 1)$-normally distributed components.

$\mathcal{N}(\boldsymbol{m}, \boldsymbol{C}) \sim \boldsymbol{m} + \mathcal{N}(\mathbf{0}, \boldsymbol{C})$, multivariate normal distribution with mean $\boldsymbol{m} \in \mathbb{R}^n$ and covariance matrix $\boldsymbol{C} \in \mathbb{R}^{n\times n}$. The matrix $\boldsymbol{C}$ is symmetric and positive definite.

$\boldsymbol{p} \in \mathbb{R}^n$, evolution path, a sequence of successive (normalized) steps, the strategy takes over a number of generations.

w_i, where $i = 1, \dots, \mu$, recombination weights, see (6).

$\boldsymbol{x}_k^{(g+1)} \in \mathbb{R}^n$, k-th offspring/individual from generation $g + 1$. We also refer to $\boldsymbol{x}^{(g+1)}$, as search point, or object parameters/variables, commonly used synonyms are candidate solution, or design variables.

$\boldsymbol{x}_{i:\lambda}^{(g+1)}$, i-th best individual out of $\boldsymbol{x}_1^{(g+1)}, \dots, \boldsymbol{x}_\lambda^{(g+1)}$, see (5). The index $i : \lambda$ denotes the index of the i-th ranked individual and $f(\boldsymbol{x}_{1:\lambda}^{(g+1)}) \le f(\boldsymbol{x}_{2:\lambda}^{(g+1)}) \le \cdots \le f(\boldsymbol{x}_{\lambda:\lambda}^{(g+1)})$, where f is the objective function to be minimized.

$\boldsymbol{y}_k^{(g+1)} = (\boldsymbol{x}_k^{(g+1)} - \boldsymbol{m}^{(g)})/\sigma^{(g)}$ corresponding to $\boldsymbol{x}_k = \boldsymbol{m} + \sigma\boldsymbol{y}_k$.

0 Preliminaries

This tutorial introduces the CMA Evolution Strategy (ES), where CMA stands for Covariance Matrix Adaptation. The CMA-ES is a stochastic method for real-parameter (continuous domain) optimization of non-linear, non-convex functions (see also Section 0.3 below).[1] We try to motivate and derive the algorithm from intuitive concepts and from requirements of non-linear, non-convex search in continuous domain. In order to refer to the described algorithm, also cite [9]. For finding a concise algorithm description go directly to Appendix A. The respective Matlab source code is given in Appendix C.

Before we start to introduce the algorithm in Sect. 1, a few required fundamentals are summed up.

[1]While CMA variants for *multi-objective* optimization and *elitistic* variants have been proposed, this tutorial is solely dedicated to single objective optimization and to non-elitistic truncation selection, also referred to as comma-selection.

0.1 Eigendecomposition of a Positive Definite Matrix

A symmetric, positive definite matrix, $\boldsymbol{C} \in \mathbb{R}^{n\times n}$, is characterized in that for all $\boldsymbol{x} \in \mathbb{R}^n \backslash \{\boldsymbol{0}\}$ holds $\boldsymbol{x}^{\mathrm{T}}\boldsymbol{C}\boldsymbol{x} > 0$. The matrix $\boldsymbol{C}$ has an orthonormal basis of eigenvectors, $\boldsymbol{B} = [\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n]$, with corresponding eigenvalues, $d_1^2, \ldots, d_n^2 > 0$.

That means for each $\boldsymbol{b}_i$ holds

$$\boldsymbol{C}\boldsymbol{b}_i = d_i^2 \boldsymbol{b}_i \quad . \tag{1}$$

The important message from (1) is that *eigenvectors are not rotated* by $\boldsymbol{C}$. This feature uniquely distinguishes eigenvectors. Because we assume the orthogonal eigenvectors to be of unit length, $\boldsymbol{b}_i^{\mathrm{T}}\boldsymbol{b}_j = \delta_{ij} = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{otherwise} \end{cases}$, and $\boldsymbol{B}^{\mathrm{T}}\boldsymbol{B} = \mathbf{I}$ (obviously this means $\boldsymbol{B}^{-1} = \boldsymbol{B}^{\mathrm{T}}$, and it follows $\boldsymbol{B}\boldsymbol{B}^{\mathrm{T}} = \mathbf{I}$). An basis of eigenvectors is practical, because for any $\boldsymbol{v} \in \mathbb{R}^n$ we can find coefficients α_i, such that $\boldsymbol{v} = \sum_i \alpha_i \boldsymbol{b}_i$, and then we have $\boldsymbol{C}\boldsymbol{v} = \sum_i d_i^2 \alpha_i \boldsymbol{b}_i$.

The eigendecomposition of $\boldsymbol{C}$ obeys

$$\boldsymbol{C} = \boldsymbol{B}\boldsymbol{D}^2\boldsymbol{B}^{\mathrm{T}} \quad , \tag{2}$$

where

$\boldsymbol{B}$ is an orthogonal matrix, $\boldsymbol{B}^{\mathrm{T}}\boldsymbol{B} = \boldsymbol{B}\boldsymbol{B}^{\mathrm{T}} = \mathbf{I}$. Columns of $\boldsymbol{B}$ form an orthonormal basis of eigenvectors.

$\boldsymbol{D}^2 = \boldsymbol{D}\boldsymbol{D} = \mathrm{diag}(d_1, \ldots, d_n)^2 = \mathrm{diag}(d_1^2, \ldots, d_n^2)$ is a diagonal matrix with eigenvalues of $\boldsymbol{C}$ as diagonal elements.

$\boldsymbol{D} = \mathrm{diag}(d_1, \ldots, d_n)$ is a diagonal matrix with square roots of eigenvalues of $\boldsymbol{C}$ as diagonal elements.

The matrix decomposition (2) is unique, apart from signs of columns of $\boldsymbol{B}$ and permutations of columns in $\boldsymbol{B}$ and $\boldsymbol{D}^2$ respectively, given all eigenvalues are different.[2]

Given the eigendecomposition (2), the inverse $\boldsymbol{C}^{-1}$ can be computed via

$$\begin{aligned} \boldsymbol{C}^{-1} &= \left(\boldsymbol{B}\boldsymbol{D}^2\boldsymbol{B}^{\mathrm{T}}\right)^{-1} \\ &= \boldsymbol{B}^{\mathrm{T}^{-1}}\boldsymbol{D}^{-2}\boldsymbol{B}^{-1} \\ &= \boldsymbol{B}\,\boldsymbol{D}^{-2}\boldsymbol{B}^{\mathrm{T}} \\ &= \boldsymbol{B}\;\mathrm{diag}\left(\frac{1}{d_1^2}, \ldots, \frac{1}{d_n^2}\right)\boldsymbol{B}^{\mathrm{T}} \quad . \end{aligned}$$

From (2) we naturally define the square root of $\boldsymbol{C}$ as

$$\boldsymbol{C}^{\frac{1}{2}} = \boldsymbol{B}\boldsymbol{D}\boldsymbol{B}^{\mathrm{T}} \tag{3}$$

and therefore

$$\begin{aligned} \boldsymbol{C}^{-\frac{1}{2}} &= \boldsymbol{B}\boldsymbol{D}^{-1}\boldsymbol{B}^{\mathrm{T}} \\ &= \boldsymbol{B}\;\mathrm{diag}\left(\frac{1}{d_1}, \ldots, \frac{1}{d_n}\right)\boldsymbol{B}^{\mathrm{T}} \end{aligned}$$

[2]Given m eigenvalues are equal, any orthonormal basis of their m-dimensional subspace can be used as column vectors. For $m > 1$ there are infinitely many such bases.

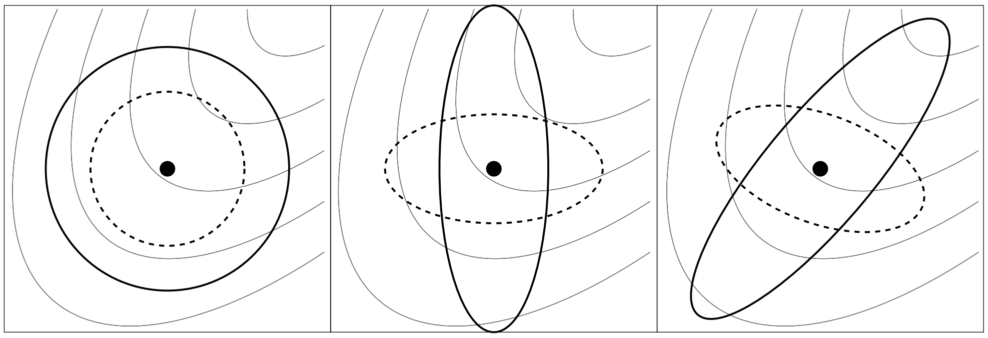

$\mathcal{N}(\mathbf{0}, \sigma^2\mathbf{I})$
$\mathcal{N}(\mathbf{0}, \boldsymbol{D}^2)$
$\mathcal{N}(\mathbf{0}, \boldsymbol{C})$

Figure 1: Ellipsoids depicting one-σ lines of equal density of six different normal distributions, where $\sigma \in \mathbb{R}_+$, $\boldsymbol{D}$ is a diagonal matrix, and $\boldsymbol{C}$ is a positive definite full covariance matrix. Thin lines depict possible objective function contour lines

0.2 The Multivariate Normal Distribution

A multivariate normal distribution, $\mathcal{N}(\boldsymbol{m}, \boldsymbol{C})$, has a unimodal, "bell-shaped" density, where the top of the bell (the modal value) corresponds to the distribution mean, $\boldsymbol{m}$. The distribution $\mathcal{N}(\boldsymbol{m}, \boldsymbol{C})$ is uniquely determined by its mean $\boldsymbol{m} \in \mathbb{R}^n$ and its symmetric and positive definite covariance matrix $\boldsymbol{C} \in \mathbb{R}^{n\times n}$. Covariance (positive definite) matrices have an appealing geometrical interpretation: they can be uniquely identified with the (hyper-)ellipsoid $\{\boldsymbol{x} \in \mathbb{R}^n \,|\, \boldsymbol{x}^{\mathrm{T}}\boldsymbol{C}^{-1}\boldsymbol{x} = 1\}$, as shown in Fig. 1. The ellipsoid is a surface of equal density of the distribution. The principal axes of the ellipsoid correspond to the eigenvectors of $\boldsymbol{C}$, the squared axes lengths correspond to the eigenvalues. The eigendecomposition is denoted by $\boldsymbol{C} = \boldsymbol{B}(\boldsymbol{D})^2\boldsymbol{B}^{\mathrm{T}}$ (see Sect. 0.1). If $\boldsymbol{D} = \sigma\mathbf{I}$, where $\sigma \in \mathbb{R}_+$ and $\mathbf{I}$ denotes the identity matrix, $\boldsymbol{C} = \sigma^2\mathbf{I}$ and the ellipsoid is isotropic (Fig. 1, left). If $\boldsymbol{B} = \mathbf{I}$, then $\boldsymbol{C} = \boldsymbol{D}^2$ is a diagonal matrix and the ellipsoid is axis parallel oriented (middle). In the coordinate system given by the columns of $\boldsymbol{B}$, the distribution $\mathcal{N}(\mathbf{0}, \boldsymbol{C})$ is always uncorrelated.

The normal distribution $\mathcal{N}(\boldsymbol{m}, \boldsymbol{C})$ can be written in different ways.

$$\begin{aligned}
\mathcal{N}(\boldsymbol{m}, \boldsymbol{C}) &\sim \boldsymbol{m} + \mathcal{N}(\mathbf{0}, \boldsymbol{C}) \\
&\sim \boldsymbol{m} + \boldsymbol{C}^{\frac{1}{2}}\mathcal{N}(\mathbf{0}, \mathbf{I}) \\
&\sim \boldsymbol{m} + \boldsymbol{B}\boldsymbol{D}\underbrace{\boldsymbol{B}^{\mathrm{T}}\mathcal{N}(\mathbf{0}, \mathbf{I})}_{\sim\mathcal{N}(\mathbf{0}, \mathbf{I})} \\
&\sim \boldsymbol{m} + \boldsymbol{B}\underbrace{\boldsymbol{D}\mathcal{N}(\mathbf{0}, \mathbf{I})}_{\sim\mathcal{N}(\mathbf{0}, \boldsymbol{D}^2)} \ ,
\end{aligned} \tag{4}$$

where "$\sim$" denotes equality in distribution, and $\boldsymbol{C}^{\frac{1}{2}} = \boldsymbol{B}\boldsymbol{D}\boldsymbol{B}^{\mathrm{T}}$. The last row can be well interpreted, from right to left

$\mathcal{N}(\mathbf{0}, \mathbf{I})$ produces an spherical (isotropic) distribution as in Fig. 1, left.

```
Initialize distribution parameters $\boldsymbol{\theta}^{(0)}$
For generation $g = 0, 1, 2, \ldots$
    Sample $\lambda$ independent points from distribution $P\left(\boldsymbol{x}|\boldsymbol{\theta}^{(g)}\right) \to \boldsymbol{x}_1, \ldots, \boldsymbol{x}_\lambda$
    Evaluate the sample $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_\lambda$ on $f$
    Update parameters $\boldsymbol{\theta}^{(g+1)} = F_\theta(\boldsymbol{\theta}^{(g)}, (\boldsymbol{x}_1, f(\boldsymbol{x}_1)), \ldots, (\boldsymbol{x}_\lambda, f(\boldsymbol{x}_\lambda)))$
    break, if termination criterion met
```

Figure 2: Randomized black box search. $f : \mathbb{R}^n \to \mathbb{R}$ is the objective function

$\boldsymbol{D}$ scales the spherical distribution within the coordinate axes as in Fig. 1, middle. $\boldsymbol{D}\mathcal{N}(\mathbf{0}, \mathbf{I}) \sim \mathcal{N}\left(\mathbf{0}, \boldsymbol{D}^2\right)$ has n independent components. The matrix $\boldsymbol{D}$ can be interpreted as (individual) step-size matrix and its diagonal entries are the standard deviations of the components.

$\boldsymbol{B}$ defines a new orientation for the ellipsoid, where the new principal axes of the ellipsoid correspond to the columns of $\boldsymbol{B}$. Note that $\boldsymbol{B}$ has $\frac{n^2-n}{2}$ degrees of freedom.

Equation (4) is useful to compute $\mathcal{N}(\boldsymbol{m}, \boldsymbol{C})$ distributed vectors, because $\mathcal{N}(\mathbf{0}, \mathbf{I})$ is a vector of independent $(0, 1)$-normally distributed numbers that can easily be realized on a computer.

0.3 Randomized Black Box Optimization

We consider the black box search scenario, where we want to *minimize an objective function* (or *cost* function or *fitness* function)

$$\begin{aligned} f \quad : \quad & \mathbb{R}^n \to \mathbb{R} \\ & \boldsymbol{x} \mapsto f(\boldsymbol{x}) \quad . \end{aligned}$$

The *objective* is to find one or more search points (candidate solutions), $\boldsymbol{x} \in \mathbb{R}^n$, with a function value, $f(\boldsymbol{x})$, as small as possible. We do not state the objective of searching for a global optimum, as this is often neither feasible nor relevant in practice. *Black box* optimization refers to the situation, where function values of evaluated search points are the only accessible information on f.[3] The search points to be evaluated can be freely chosen. We define the *search costs* as the number of executed function evaluations, in other words the amount of information we needed to aquire from f[4]. Any performance measure must consider the search costs *together* with the achieved objective function value.[5]

A randomized black box search algorithm is outlined in Fig. 2. In the CMA Evolution Strategy the search distribution, P, is a multivariate normal distribution. Given all variances and covariances, the normal distribution has the largest entropy of all distributions in $\mathbb{R}^n$.

[3]Knowledge about the underlying optimization problem might well enter the composition of f and the chosen problem *encoding*.

[4]Also f is sometimes denoted as *cost function*, but it should not to be confused with the *search costs*.

[5]A performance measure can be obtained from a number of trials as, for example, the mean number of function evaluations to reach a given function value, or the median best function value obtained after a given number of function evaluations.

Furthermore, coordinate directions are not distinguished in any way. Both makes the normal distribution a particularly attractive candidate for randomized search.

Randomized search algorithms are regarded to be robust in a rugged search landscape, which can comprise discontinuities, (sharp) ridges, or local optima. The covariance matrix adaptation (CMA) in particular is designed to tackle, additionally, ill-conditioned and non-separable[6] problems.

0.4 Hessian and Covariance Matrices

We consider the convex-quadratic objective function $f_{\boldsymbol{H}} : \boldsymbol{x} \mapsto \frac{1}{2}\boldsymbol{x}^{\mathrm{T}}\boldsymbol{H}\boldsymbol{x}$, where the Hessian matrix $\boldsymbol{H} \in \mathbb{R}^{n\times n}$ is a positive definite matrix. Given a search distribution $\mathcal{N}(\boldsymbol{m}, \boldsymbol{C})$, there is a close relation between $\boldsymbol{H}$ and $\boldsymbol{C}$: Setting $\boldsymbol{C} = \boldsymbol{H}^{-1}$ on $f_{\boldsymbol{H}}$ is equivalent to optimizing the isotropic function $f_{\mathrm{sphere}}(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^{\mathrm{T}}\boldsymbol{x} = \frac{1}{2}\sum_i x_i^2$ (where $\boldsymbol{H} = \mathbf{I}$) with $\boldsymbol{C} = \mathbf{I}$.[7] That is, on convex-quadratic objective functions, setting the covariance matrix of the search distribution to the inverse Hessian matrix is equivalent to rescaling the ellipsoid function into a spherical one. Consequently, we assume that the optimal covariance matrix equals to the inverse Hessian matrix, up to a constant factor.[8] Furthermore, choosing a covariance matrix or choosing a respective affine linear transformation of the search space (*i.e.* of $\boldsymbol{x}$) is equivalent [8], because for any full rank $n \times n$-matrix $\boldsymbol{A}$ we find a positive definite Hessian such that $\frac{1}{2}(\boldsymbol{A}\boldsymbol{x})^{\mathrm{T}}\boldsymbol{A}\boldsymbol{x} = \frac{1}{2}\boldsymbol{x}^{\mathrm{T}}\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{x} = \frac{1}{2}\boldsymbol{x}^{\mathrm{T}}\boldsymbol{H}\boldsymbol{x}$.

The final **objective** of covariance matrix adaptation is to closely *approximate the contour lines of the objective function f*. On convex-quadratic functions this amounts to approximating the inverse Hessian matrix, similar to a quasi-Newton method.

In Fig. 1 the solid-line distribution in the right figure follows the objective function contours most suitably, and it is easy to foresee that it will aid to approach the optimum the most.

The **condition number** of a positive definite matrix $\boldsymbol{A}$ is defined via the Euclidean norm: $\mathtt{cond}(\boldsymbol{A}) \overset{\mathrm{def}}{=} \|\boldsymbol{A}\| \times \|\boldsymbol{A}^{-1}\|$, where $\|\boldsymbol{A}\| = \sup_{\|\boldsymbol{x}\|=1} \|\boldsymbol{A}\boldsymbol{x}\|$. For a positive definite (Hessian or covariance) matrix $\boldsymbol{A}$ holds $\|\boldsymbol{A}\| = \lambda_{\max}$ and $\mathtt{cond}(\boldsymbol{A}) = \frac{\lambda_{\max}}{\lambda_{\min}} \geq 1$, where $\lambda_{\max}$ and $\lambda_{\min}$ are the largest and smallest eigenvalue of $\boldsymbol{A}$.

1 Basic Equation: Sampling

In the CMA Evolution Strategy, a population of new search points (individuals, offspring) is generated by sampling a multivariate normal distribution.[9] The basic equation for sampling the search points, for generation number $g = 0, 1, 2, \ldots$, reads[10]

$$\boxed{\boldsymbol{x}_k^{(g+1)} \sim \boldsymbol{m}^{(g)} + \sigma^{(g)}\mathcal{N}\left(\mathbf{0}, \boldsymbol{C}^{(g)}\right) \qquad \text{for } k = 1, \ldots, \lambda \qquad (5)}$$

[6]An n-dimensional *separable* problem can be solved by solving n 1-dimensional problems separately, which is a far easier task.

[7]Also the initial mean value $\boldsymbol{m}$ has to be transformed accordingly.

[8]Even though there is good intuition and strong empirical evidence for this statement, a rigorous proof is missing.

[9]Recall that, given all (co-)variances, the normal distribution has the largest entropy of all distributions in $\mathbb{R}^n$.

[10]Framed equations belong to the final algorithm of a CMA Evolution Strategy.

where

$\sim$ denotes the same distribution on the left and right side.

$\mathcal{N}(\mathbf{0}, \boldsymbol{C}^{(g)})$ is a multivariate normal distribution with zero mean and covariance matrix $\boldsymbol{C}^{(g)}$, see Sect. 0.2. It holds $\boldsymbol{m}^{(g)} + \sigma^{(g)}\mathcal{N}(\mathbf{0}, \boldsymbol{C}^{(g)}) \sim \mathcal{N}\big(\boldsymbol{m}^{(g)}, (\sigma^{(g)})^2\boldsymbol{C}^{(g)}\big)$.

$\boldsymbol{x}_k^{(g+1)} \in \mathbb{R}^n$, k-th offspring (individual, search point) from generation $g+1$.

$\boldsymbol{m}^{(g)} \in \mathbb{R}^n$, mean value of the search distribution at generation g.

$\sigma^{(g)} \in \mathbb{R}_+$, "overall" standard deviation, step-size, at generation g.

$\boldsymbol{C}^{(g)} \in \mathbb{R}^{n\times n}$, covariance matrix at generation g. Up to the scalar factor ${\sigma^{(g)}}^2$, $\boldsymbol{C}^{(g)}$ is the covariance matrix of the search distribution.

$\lambda \geq 2$, population size, sample size, number of offspring.

To define the complete iteration step, the remaining question is, how to calculate $\boldsymbol{m}^{(g+1)}$, $\boldsymbol{C}^{(g+1)}$, and $\sigma^{(g+1)}$ for the next generation $g+1$. The next three sections will answer these questions, respectively. An algorithm summary with all parameter settings and MATLAB source code is given in Appendix A and C, respectively.

2 Selection and Recombination: Moving the Mean

The new mean $\boldsymbol{m}^{(g+1)}$ of the search distribution is a *weighted average of* μ *selected points* from the sample $\boldsymbol{x}_1^{(g+1)}, \ldots, \boldsymbol{x}_\lambda^{(g+1)}$:

$$\boxed{\boldsymbol{m}^{(g+1)} = \sum_{i=1}^{\mu} w_i\, \boldsymbol{x}_{i:\lambda}^{(g+1)} \qquad (6)}$$

$$\sum_{i=1}^{\mu} w_i = 1, \qquad w_1 \geq w_2 \geq \cdots \geq w_\mu > 0 \tag{7}$$

where

$\mu \leq \lambda$ is the parent population size, *i.e.* the number of selected points.

$w_{i=1\ldots\mu} \in \mathbb{R}_+$, positive weight coefficients for recombination. For $w_{i=1\ldots\mu} = 1/\mu$, Equation (6) calculates the mean value of μ selected points.

$\boldsymbol{x}_{i:\lambda}^{(g+1)}$, i-th best individual out of $\boldsymbol{x}_1^{(g+1)}, \ldots, \boldsymbol{x}_\lambda^{(g+1)}$ from (5). The index $i:\lambda$ denotes the index of the i-th ranked individual and $f(\boldsymbol{x}_{1:\lambda}^{(g+1)}) \leq f(\boldsymbol{x}_{2:\lambda}^{(g+1)}) \leq \cdots \leq f(\boldsymbol{x}_{\lambda:\lambda}^{(g+1)})$, where f is the objective function to be minimized.

Equation (6) implements *truncation selection* by choosing $\mu < \lambda$ out of λ offspring points. Assigning *different* weights w_i must also be interpreted as a selection mechanism. Equation (6) implements *weighted intermediate recombination* by taking $\mu > 1$ individuals into account for a weighted average.

The measure

$$\mu_{\text{eff}} = \left(\frac{\|\boldsymbol{w}\|_1}{\|\boldsymbol{w}\|_2}\right)^2 = \frac{\|\boldsymbol{w}\|_1^2}{\|\boldsymbol{w}\|_2^2} = \frac{1}{\|\boldsymbol{w}\|_2^2} = \left(\sum_{i=1}^{\mu} w_i^2\right)^{-1} \tag{8}$$

will be repeatedly used in the following and can be paraphrased as *variance effective selection mass*. From the definition of w_i in (7) we derive $1 \leq \mu_{\text{eff}} \leq \mu$, and $\mu_{\text{eff}} = \mu$ for equal recombination weights, *i.e.* $w_i = 1/\mu$ for all $i = 1 \ldots \mu$. Usually, $\mu_{\text{eff}} \approx \lambda/4$ indicates a reasonable setting of w_i. A typical setting could be $w_i \propto \mu - i + 1$, and $\mu \approx \lambda/2$.

3 Adapting the Covariance Matrix

In this section, the update of the covariance matrix, $\boldsymbol{C}$, is derived. We will start out estimating the covariance matrix from a single population of one generation (Sect. 3.1). For small populations this estimation is unreliable and an adaptation procedure has to be invented (rank-μ-update, Sect. 3.2). In the limit case only a single point can be used to update (adapt) the covariance matrix at each generation (rank-one-update, Sect. 3.3). The adaptation can be enhanced by exploiting dependencies between successive steps applying cumulation (Sect. 3.3.2). Finally we combine the rank-μ and rank-one updating methods (Sect. 3.4).

3.1 Estimating the Covariance Matrix From Scratch

For the moment we assume that the population contains enough information to reliably estimate a covariance matrix from the population.[11] For the sake of convenience we assume $\sigma^{(g)} = 1$ (see (5)) in this section. For $\sigma^{(g)} \neq 1$ the formulae hold except for a constant factor.

We can (re-)estimate the original covariance matrix $\boldsymbol{C}^{(g)}$ using the sampled population from (5), $\boldsymbol{x}_1^{(g+1)} \ldots \boldsymbol{x}_\lambda^{(g+1)}$, via the empirical covariance matrix

$$\boldsymbol{C}_{\text{emp}}^{(g+1)} = \frac{1}{\lambda - 1}\sum_{i=1}^{\lambda}\left(\boldsymbol{x}_i^{(g+1)} - \frac{1}{\lambda}\sum_{j=1}^{\lambda}\boldsymbol{x}_j^{(g+1)}\right)\left(\boldsymbol{x}_i^{(g+1)} - \frac{1}{\lambda}\sum_{j=1}^{\lambda}\boldsymbol{x}_j^{(g+1)}\right)^{\mathrm{T}} . \tag{9}$$

The empirical covariance matrix $\boldsymbol{C}_{\text{emp}}^{(g+1)}$ is an unbiased estimator of $\boldsymbol{C}^{(g)}$: assuming the $\boldsymbol{x}_i^{(g+1)}, i = 1 \ldots \lambda$, to be random variables (rather than a realized sample), we have that $\mathsf{E}\left[\boldsymbol{C}_{\text{emp}}^{(g+1)} \,\middle|\, \boldsymbol{C}^{(g)}\right] = \boldsymbol{C}^{(g)}$. Consider now a slightly different approach to get an estimator for $\boldsymbol{C}^{(g)}$.

$$\boldsymbol{C}_{\lambda}^{(g+1)} = \frac{1}{\lambda}\sum_{i=1}^{\lambda}\left(\boldsymbol{x}_i^{(g+1)} - \boldsymbol{m}^{(g)}\right)\left(\boldsymbol{x}_i^{(g+1)} - \boldsymbol{m}^{(g)}\right)^{\mathrm{T}} \tag{10}$$

[11] To re-estimate the covariance matrix, $\boldsymbol{C}$, from a $\mathcal{N}(\boldsymbol{0}, \mathbf{I})$ distributed sample such that $\texttt{cond}(\boldsymbol{C}) < 10$ a sample size $\lambda \geq 4n$ is needed, as can be observed in numerical experiments.

Also the matrix $\boldsymbol{C}_\lambda^{(g+1)}$ is an unbiased estimator of $\boldsymbol{C}^{(g)}$. The remarkable difference between (9) and (10) is the reference mean value. For $\boldsymbol{C}_{\mathrm{emp}}^{(g+1)}$ it is the mean of the *actually realized* sample. For $\boldsymbol{C}_\lambda^{(g+1)}$ it is the *true* mean value, $\boldsymbol{m}^{(g)}$, of the sampled distribution (see (5)). Therefore, the estimators $\boldsymbol{C}_{\mathrm{emp}}^{(g+1)}$ and $\boldsymbol{C}_\lambda^{(g+1)}$ can be interpreted differently: while $\boldsymbol{C}_{\mathrm{emp}}^{(g+1)}$ estimates the distribution variance *within the sampled points*, $\boldsymbol{C}_\lambda^{(g+1)}$ estimates variances of sampled *steps*, $\boldsymbol{x}_i^{(g+1)} - \boldsymbol{m}^{(g)}$.

A minor difference between (9) and (10) is the different normalizations $\frac{1}{\lambda-1}$ versus $\frac{1}{\lambda}$, necessary to get an unbiased estimator in both cases. In (9) one degree of freedom is already taken by the inner summand. In order to get a maximum likelihood estimator in both cases $\frac{1}{\lambda}$ must be used.

Equation (10) re-estimates *the original* covariance matrix. To "estimate" a "better" covariance matrix (10) is modified and the same, *weighted selection* mechanism as in (6) is used [11].

$$\boldsymbol{C}_\mu^{(g+1)} = \sum_{i=1}^{\mu} w_i \left(\boldsymbol{x}_{i:\lambda}^{(g+1)} - \boldsymbol{m}^{(g)}\right)\left(\boldsymbol{x}_{i:\lambda}^{(g+1)} - \boldsymbol{m}^{(g)}\right)^{\mathrm{T}} \tag{11}$$

The matrix $\boldsymbol{C}_\mu^{(g+1)}$is an estimator for the distribution of *selected steps*, just as $\boldsymbol{C}_\lambda^{(g+1)}$ is an estimator of the original distribution of steps before selection. Sampling from $\boldsymbol{C}_\mu^{(g+1)}$ tends to reproduce selected, *i.e. successful* steps, giving a justification for what a "better" covariance matrix means.

Following [9], we compare (11) with the Estimation of Multivariate Normal Algorithm EMNA$_{global}$ [18, 19]. The covariance matrix in EMNA$_{global}$ reads, similar to (9),

$$\boldsymbol{C}_{\mathrm{EMNA}_{global}}^{(g+1)} = \frac{1}{\mu}\sum_{i=1}^{\mu} \left(\boldsymbol{x}_{i:\lambda}^{(g+1)} - \boldsymbol{m}^{(g+1)}\right)\left(\boldsymbol{x}_{i:\lambda}^{(g+1)} - \boldsymbol{m}^{(g+1)}\right)^{\mathrm{T}}, \tag{12}$$

where $\boldsymbol{m}^{(g+1)} = \frac{1}{\mu}\sum_{i=1}^{\mu} \boldsymbol{x}_{i:\lambda}^{(g+1)}$. Similarly, applying the so-called Cross-Entropy method to continuous domain optimization [21] yields the covariance matrix $\frac{\mu}{\mu-1}\boldsymbol{C}_{\mathrm{EMNA}_{global}}^{(g+1)}$, *i.e.* the *unbiased* empirical covariance matrix of the μ best points. In both cases the subtle, but most important difference to (11) is, again, the choice of the reference mean value.[12] Equation (12) estimates the variance *within* the selected population while (11) estimates selected steps. Equation (12) reveals always smaller variances than (11), because its reference mean value is the minimizer for the variances. Moreover, in most conceivable selection situations (12) decreases the variances compared to $\boldsymbol{C}^{(g)}$.

Figure 3 demonstrates the estimation results on *a linear* objective function for $\lambda = 150$, $\mu = 50$, and $w_i = 1/\mu$. Equation (11) geometrically increases the expected variance in direction of the gradient (where the selection takes place, here the diagonal), given ordinary settings for parent number μ and recombination weights $w_1, \dots, w_\mu$. Equation (12) always decreases the variance in gradient direction geometrically fast! Therefore, (12) is highly susceptible to premature convergence, in particular with small parent populations, where the population cannot be expected to bracket the optimum at any time. However, for large values of μ in large populations with large initial variances, the impact of the different reference mean value can become marginal.

[12]Taking a weighted sum, $\sum_{i=1}^{\mu} w_i \dots$, instead of the mean, $\frac{1}{\mu}\sum_{i=1}^{\mu} \dots$, is an appealing, but less important, difference.

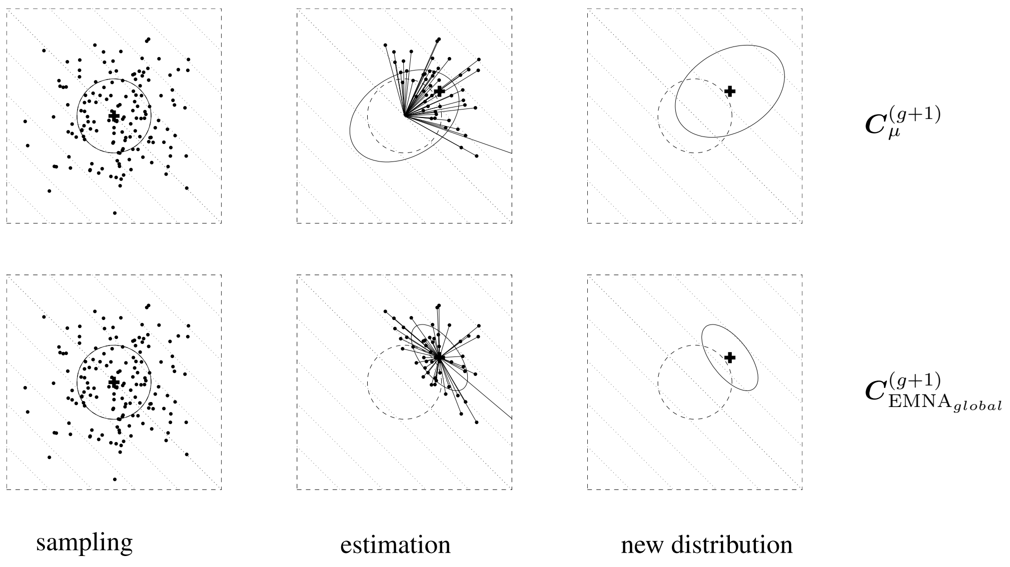


Figure 3: Estimation of the covariance matrix on $f_{\mathrm{linear}}(\boldsymbol{x}) = -\sum_{i=1}^{2} x_i$ to be minimized. Contour lines (*dotted*) indicate that the strategy should move toward the upper right corner. **Above**: estimation of $\boldsymbol{C}_\mu^{(g+1)}$ according to (11), where $w_i = 1/\mu$. **Below**: estimation of $\boldsymbol{C}_{\mathrm{EMNA}_{global}}^{(g+1)}$ according to (12). Left: sample of $\lambda = 150\ \mathcal{N}(\mathbf{0}, \mathbf{I})$ distributed points. Middle: the $\mu = 50$ selected points (*dots*) determining the entries for the estimation equation (*solid straight lines*). Right: search distribution of the next generation (*solid ellipsoids*). Given $w_i = 1/\mu$, estimation via $\boldsymbol{C}_\mu^{(g+1)}$ *increases* the expected variance in gradient direction for all $\mu < \lambda/2$, while estimation via $\boldsymbol{C}_{\mathrm{EMNA}_{global}}^{(g+1)}$ *decreases* this variance for any $\mu < \lambda$ geometrically fast

In order to ensure with (5), (6), and (11), that $\boldsymbol{C}_\mu^{(g+1)}$ is a *reliable* estimator, the variance effective selection mass μ_{eff} (cf. (8)) must be large enough: getting condition numbers (cf. Sect. 0.4) smaller than ten for $\boldsymbol{C}_\mu^{(g)}$ on $f_{\mathrm{sphere}}(\boldsymbol{x}) = \sum_{i=1}^{n} x_i^2$, requires $\mu_{\mathrm{eff}} \approx 10n$. The next step is to circumvent this restriction on μ_{eff}.

3.2 Rank-μ-Update

To achieve *fast* search (opposite to *more robust* or *more global* search), e.g. competitive performance on f_{sphere}, the population size λ must be small. Because $\mu_{\mathrm{eff}} \approx \lambda/4$ also μ_{eff} must be small and we may assume, e.g., $\mu_{\mathrm{eff}} \leq 1 + \ln n$. Then, it is not possible to get a *reliable* estimator for a good covariance matrix from (11). As a remedy, information from previous generations is used additionally. For example, after a sufficient number of generations, the mean of the estimated covariance matrices from all generations,

$$\boldsymbol{C}^{(g+1)} = \frac{1}{g+1}\sum_{i=0}^{g}\frac{1}{{\sigma^{(i)}}^2}\boldsymbol{C}_\mu^{(i+1)} \tag{13}$$

becomes a reliable estimator for the selected steps. To make $\boldsymbol{C}_\mu^{(g)}$ from different generations comparable, the different $\sigma^{(i)}$ are incorporated. (Assuming $\sigma^{(i)} = 1$, (13) resembles the covariance matrix from the Estimation of Multivariate Normal Algorithm EMNA$_i$ [19].)

In (13), all generation steps have the same weight. To assign recent generations a higher weight, exponential smoothing is introduced. Choosing $\boldsymbol{C}^{(0)} = \mathbf{I}$ to be the unity matrix and a learning rate $0 < c_\mu \leq 1$, then $\boldsymbol{C}^{(g+1)}$ reads

$$\begin{aligned}
\boldsymbol{C}^{(g+1)} &= (1-c_\mu)\boldsymbol{C}^{(g)} + c_\mu\frac{1}{{\sigma^{(g)}}^2}\boldsymbol{C}_\mu^{(g+1)} \\
&= (1-c_\mu)\boldsymbol{C}^{(g)} + c_\mu\sum_{i=1}^{\mu} w_i\, \boldsymbol{y}_{i:\lambda}^{(g+1)}{\boldsymbol{y}_{i:\lambda}^{(g+1)}}^{\mathrm{T}} \\
&= {\boldsymbol{C}^{(g)}}^{1/2}\left(\mathbf{I} + c_\mu\sum_{i=1}^{\mu} w_i\left(\boldsymbol{z}_{i:\lambda}^{(g+1)}{\boldsymbol{z}_{i:\lambda}^{(g+1)}}^{\mathrm{T}} - \mathbf{I}\right)\right){\boldsymbol{C}^{(g)}}^{1/2}
\end{aligned} \tag{14}$$

where

$c_\mu \leq 1$ learning rate for updating the covariance matrix. For $c_\mu = 1$, no prior information is retained and $\boldsymbol{C}^{(g+1)} = \frac{1}{{\sigma^{(g)}}^2}\boldsymbol{C}_\mu^{(g+1)}$. For $c_\mu = 0$, no learning takes place and $\boldsymbol{C}^{(g+1)} = \boldsymbol{C}^{(0)}$. Here, $c_\mu \approx \min(1, \mu_{\mathrm{eff}}/n^2)$ is a reasonably choice.

$\boldsymbol{y}_{i:\lambda}^{(g+1)} = (\boldsymbol{x}_{i:\lambda}^{(g+1)} - \boldsymbol{m}^{(g)})/\sigma^{(g)}$.

$\boldsymbol{z}_{i:\lambda}^{(g+1)} = {\boldsymbol{C}^{(g)}}^{-1/2}\boldsymbol{y}_{i:\lambda}^{(g+1)}$ is the mutation vector expressed in the unique coordinate system where the sampling is isotropic and the respective coordinate system transformation does not rotate the original principal axes of the distribution.

This covariance matrix update is called rank-μ-update [13], because the sum of outer products in (14) is of rank $\min(\mu, n)$ (with probability one). Note that this sum can even consist of a single term, if $\mu = 1$.

The number $1/c_\mu$ is the *backward time horizon* that contains roughly 63% of the overall weight.

Because (14) expands to the weighted sum

$$\boldsymbol{C}^{(g+1)} = (1-c_\mu)^{g+1}\boldsymbol{C}^{(0)} + c_\mu\sum_{i=0}^{g}(1-c_\mu)^{g-i}\frac{1}{{\sigma^{(i)}}^2}\,\boldsymbol{C}_\mu^{(i+1)}\ , \tag{15}$$

the backward time horizon, Δg, where about 63% of the overall weight is summed up, is defined by

$$c_\mu\sum_{i=g+1-\Delta g}^{g}(1-c_\mu)^{g-i} \approx 0.63 \approx 1 - \frac{1}{\mathrm{e}}\ . \tag{16}$$

Resolving the sum yields

$$(1 - c_\mu)^{\Delta g} \approx \frac{1}{\mathrm{e}} \quad , \tag{17}$$

and resolving for Δg, using the Taylor approximation for $\ln$, yields

$$\Delta g \approx \frac{1}{c_\mu} \quad . \tag{18}$$

That is, approximately 37% of the information in $\boldsymbol{C}^{(g+1)}$ is older than $1/c_\mu$ generations, and, according to (17), the original weight is reduced by a factor of 0.37 after approximately $1/c_\mu$ generations.[13]

The choice of c_μ is crucial. Small values lead to slow learning, too large values lead to a failure, because the covariance matrix degenerates. Fortunately, a good setting seems to be largely independent of the function to be optimized.[14] A first order approximation for a good choice is $c_\mu \approx \mu_{\text{eff}}/n^2$. Therefore, the characteristic time horizon for (14) is roughly n^2/μ_{eff}.

Even for the learning rate $c_\mu = 1$, adapting the covariance matrix cannot be accomplished within one generation. The effect of the original sample distribution does not vanish until a sufficient number of generations. Assuming fixed search costs (number of function evaluations), a small population size λ allows a larger number of generations and therefore usually leads to a faster adaptation of the covariance matrix.

3.3 Rank-One-Update

In Section 3.1 we estimated the complete covariance matrix from scratch, using all selected steps from a *single generation*. We now take precisely the opposite viewpoint. We will repeatedly *update* the covariance matrix in the generation sequence using a *single selected step* only. First, this perspective will give a justification of the adaptation rule (14). Second, we will introduce the so-called evolution path that is finally used for a rank-one update of the covariance matrix.

3.3.1 A Different Viewpoint

We consider a specific method to produce n-dimensional normal distributions with zero mean. Let the vectors $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_{g_0} \in \mathbb{R}^n$, $g_0 \geq n$, span $\mathbb{R}^n$ and let $\mathcal{N}(0,1)$ denote independent $(0,1)$-normally distributed random numbers, then

$$\mathcal{N}(0,1)\,\boldsymbol{y}_1 + \cdots + \mathcal{N}(0,1)\,\boldsymbol{y}_{g_0} \quad \sim \quad \mathcal{N}\left(\mathbf{0}, \sum_{i=1}^{g_0} \boldsymbol{y}_i \boldsymbol{y}_i^{\mathrm{T}}\right) \tag{19}$$

is a normally distributed random vector with zero mean and covariance matrix $\sum_{i=1}^{g_0} \boldsymbol{y}_i \boldsymbol{y}_i^{\mathrm{T}}$. The random vector (19) is generated by adding "line-distributions" $\mathcal{N}(0,1)\,\boldsymbol{y}_i$. The singular distribution $\mathcal{N}(0,1)\,\boldsymbol{y}_i \sim \mathcal{N}(\mathbf{0}, \boldsymbol{y}_i\boldsymbol{y}_i^{\mathrm{T}})$ generates the vector $\boldsymbol{y}_i$ with maximum likelihood considering all normal distributions with zero mean.

[13]This can be shown more easily, because $(1-c_\mu)^g = \exp\ln(1-c_\mu)^g = \exp(g\ln(1-c_\mu)) \approx \exp(-gc_\mu)$ for small c_μ, and for $g \approx 1/c_\mu$ we get immediately $(1-c_\mu)^g \approx \exp(-1)$.

[14]We use the sphere model $f_{\text{sphere}}(\boldsymbol{x}) = \sum_i x_i^2$ to empirically find a good setting for the parameter c_μ, dependent on n and μ_{eff}. The found setting was applicable to any non-noisy objective function we tried so far.

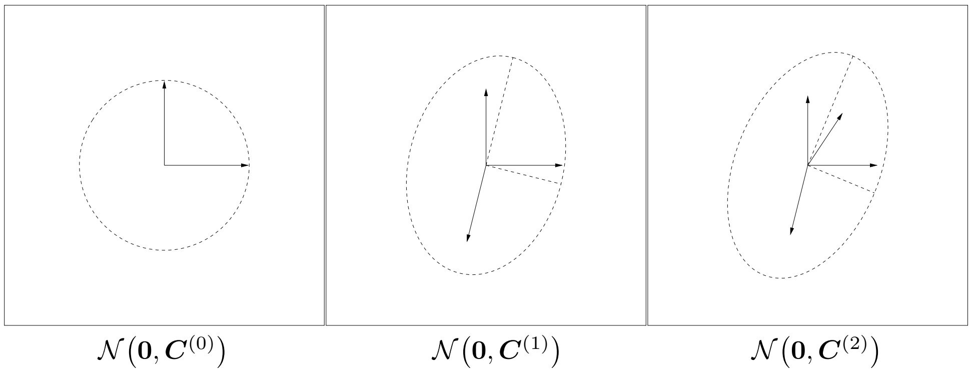

$\mathcal{N}(\mathbf{0}, \boldsymbol{C}^{(0)})$
$\mathcal{N}(\mathbf{0}, \boldsymbol{C}^{(1)})$
$\mathcal{N}(\mathbf{0}, \boldsymbol{C}^{(2)})$

Figure 4: Change of the distribution according to the covariance matrix update (20). Left: vectors $\boldsymbol{e}_1$ and $\boldsymbol{e}_2$, and $\boldsymbol{C}^{(0)} = \mathbf{I} = \boldsymbol{e}_1\boldsymbol{e}_1^{\mathrm{T}} + \boldsymbol{e}_2\boldsymbol{e}_2^{\mathrm{T}}$. Middle: vectors $0.91\,\boldsymbol{e}_1$, $0.91\,\boldsymbol{e}_2$, and $0.41\,\boldsymbol{y}_1$ (the coefficients deduce from $c_1 = 0.17$), and $\boldsymbol{C}^{(1)} = (1-c_1)\,\mathbf{I} + c_1\,\boldsymbol{y}_1\boldsymbol{y}_1^{\mathrm{T}}$, where $\boldsymbol{y}_1 = \binom{-0.59}{-2.2}$. The distribution ellipsoid is elongated into the direction of $\boldsymbol{y}_1$, and therefore increases the likelihood of $\boldsymbol{y}_1$. Right: $\boldsymbol{C}^{(2)} = (1-c_1)\,\boldsymbol{C}^{(1)} + c_1\,\boldsymbol{y}_2\boldsymbol{y}_2^{\mathrm{T}}$, where $\boldsymbol{y}_2 = \binom{0.97}{1.5}$.

The line distribution that generates a vector $\boldsymbol{y}$ with the maximum likelihood must "live" on a line that includes $\boldsymbol{y}$, and therefore the distribution must obey $\mathcal{N}(0,1)\sigma\boldsymbol{y} \sim \mathcal{N}(0, \sigma^2\boldsymbol{y}\boldsymbol{y}^{\mathrm{T}})$. Any other line distribution with zero mean cannot generate $\boldsymbol{y}$ at all. Choosing σ reduces to choosing the maximum likelihood of $\|\boldsymbol{y}\|$ for the one-dimensional gaussian $\mathcal{N}(0, \sigma^2\|\boldsymbol{y}\|^2)$, which is $\sigma = 1$.

The covariance matrix $\boldsymbol{y}\boldsymbol{y}^{\mathrm{T}}$ has rank one, its only eigenvectors are $\mathbb{R}_{\setminus 0} \times \boldsymbol{y}$ with eigenvalue $\|\boldsymbol{y}\|^2$. Using equation (19), any normal distribution can be realized if $\boldsymbol{y}_i$ are chosen appropriately. For example, (19) resembles (4) with $\boldsymbol{m} = \mathbf{0}$, using the orthogonal eigenvectors $\boldsymbol{y}_i = d_{ii}\boldsymbol{b}_i$, for $i = 1, \dots, n$, where $\boldsymbol{b}_i$ are the columns of $\boldsymbol{B}$. In general, the vectors $\boldsymbol{y}_i$ need not to be eigenvectors of the covariance matrix (and usually are not).

Considering (19) and a slight simplification of (14), we try to gain insight into the adaptation rule for the covariance matrix. Let the sum in (14) consist of a single summand only (*e.g.* $\mu = 1$), and let $\boldsymbol{y}_{g+1} = \frac{\boldsymbol{x}_{1:\lambda}^{(g+1)} - \boldsymbol{m}^{(g)}}{\sigma^{(g)}}$. Then, the rank-one update for the covariance matrix reads

$$\boldsymbol{C}^{(g+1)} = (1-c_1)\boldsymbol{C}^{(g)} + c_1\,\boldsymbol{y}_{g+1}{\boldsymbol{y}_{g+1}}^{\mathrm{T}} \qquad (20)$$

The right summand is of rank one and adds the maximum likelihood term for $\boldsymbol{y}_{g+1}$ into the covariance matrix $\boldsymbol{C}^{(g)}$. Therefore the probability to generate $\boldsymbol{y}_{g+1}$ in the next generation increases.

An example of the first two iteration steps of (20) is shown in **Figure 4**. The distribution $\mathcal{N}(\mathbf{0}, \boldsymbol{C}^{(1)})$ tends to reproduce $\boldsymbol{y}_1$ with a larger probability than the initial distribution $\mathcal{N}(\mathbf{0}, \mathbf{I})$; the distribution $\mathcal{N}(\mathbf{0}, \boldsymbol{C}^{(2)})$ tends to reproduce $\boldsymbol{y}_2$ with a larger probability than $\mathcal{N}(\mathbf{0}, \boldsymbol{C}^{(1)})$, and so forth. When $\boldsymbol{y}_1, \dots, \boldsymbol{y}_g$ denote the formerly selected, favorable steps, $\mathcal{N}(\mathbf{0}, \boldsymbol{C}^{(g)})$ tends to reproduce these steps. The process leads to an alignment of the search distribution $\mathcal{N}(\mathbf{0}, \boldsymbol{C}^{(g)})$ to the distribution of the selected steps. If both distributions become alike, as under random selection, in expectation no further change of the covariance matrix takes place [7].

3.3.2 Cumulation: Utilizing the Evolution Path

We have used the selected steps, $\boldsymbol{y}_{i:\lambda}^{(g+1)} = (\boldsymbol{x}_{i:\lambda}^{(g+1)} - \boldsymbol{m}^{(g)})/\sigma^{(g)}$, to update the covariance matrix in (14) and (20). Because $\boldsymbol{y}\boldsymbol{y}^{\mathrm{T}} = -\boldsymbol{y}(-\boldsymbol{y})^{\mathrm{T}}$, *the sign of the steps is irrelevant* for the update of the covariance matrix—that is, the sign information is not used for calculating $\boldsymbol{C}^{(g+1)}$. To exploit the sign information, the so-called *evolution path* is introduced [14, 16].

We call a sequence of successive steps, the strategy takes over a number of generations, an evolution path. An evolution path can be expressed by a sum of consecutive steps. This summation is referred to as *cumulation*. To construct an evolution path, the step-size σ is disregarded. For example, an evolution path of three steps of the distribution mean $\boldsymbol{m}$ can be constructed by the sum

$$\frac{\boldsymbol{m}^{(g+1)} - \boldsymbol{m}^{(g)}}{\sigma^{(g)}} + \frac{\boldsymbol{m}^{(g)} - \boldsymbol{m}^{(g-1)}}{\sigma^{(g-1)}} + \frac{\boldsymbol{m}^{(g-1)} - \boldsymbol{m}^{(g-2)}}{\sigma^{(g-2)}} \quad . \tag{21}$$

In practice, to construct the evolution path, $\boldsymbol{p}_{\mathrm{c}} \in \mathbb{R}^n$, we use exponential smoothing as in (14), and start with $\boldsymbol{p}_{\mathrm{c}}^{(0)} = \boldsymbol{0}$.[15]

$$\boxed{\boldsymbol{p}_{\mathrm{c}}^{(g+1)} = (1 - c_{\mathrm{c}})\boldsymbol{p}_{\mathrm{c}}^{(g)} + \sqrt{c_{\mathrm{c}}(2 - c_{\mathrm{c}})\mu_{\mathrm{eff}}}\,\frac{\boldsymbol{m}^{(g+1)} - \boldsymbol{m}^{(g)}}{\sigma^{(g)}} \tag{22}}$$

where

$\boldsymbol{p}_{\mathrm{c}}^{(g)} \in \mathbb{R}^n$, evolution path at generation g.

$c_{\mathrm{c}} \leq 1$. Again, $1/c_{\mathrm{c}}$ is the backward time horizon of the evolution path $\boldsymbol{p}_{\mathrm{c}}$ that contains roughly 63% of the overall weight (compare derivation of (18)). A time horizon between $\sqrt{n}$ and n is reasonable.

The factor $\sqrt{c_{\mathrm{c}}(2 - c_{\mathrm{c}})\mu_{\mathrm{eff}}}$ is a normalization constant for $\boldsymbol{p}_{\mathrm{c}}$. For $c_{\mathrm{c}} = 1$ and $\mu_{\mathrm{eff}} = 1$, the factor reduces to one, and $\boldsymbol{p}_{\mathrm{c}}^{(g+1)} = (\boldsymbol{x}_{1:\lambda}^{(g+1)} - \boldsymbol{m}^{(g)})/\sigma^{(g)}$.

The factor $\sqrt{c_{\mathrm{c}}(2 - c_{\mathrm{c}})\mu_{\mathrm{eff}}}$ is chosen, such that

$$\boldsymbol{p}_{\mathrm{c}}^{(g+1)} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{C}) \tag{23}$$

if

$$\boldsymbol{p}_{\mathrm{c}}^{(g)} \sim \frac{\boldsymbol{x}_{i:\lambda}^{(g+1)} - \boldsymbol{m}^{(g)}}{\sigma^{(g)}} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{C}) \quad \text{for all } i = 1, \ldots, \mu \quad . \tag{24}$$

To derive (23) from (24) and (22) remark that

$$(1 - c_{\mathrm{c}})^2 + \sqrt{c_{\mathrm{c}}(2 - c_{\mathrm{c}})}^2 = 1 \quad \text{and} \quad \sum_{i=1}^{\mu} w_i \mathcal{N}_i(\boldsymbol{0}, \boldsymbol{C}) \sim \frac{1}{\sqrt{\mu_{\mathrm{eff}}}}\mathcal{N}(\boldsymbol{0}, \boldsymbol{C}) \quad . \tag{25}$$

The (rank-one) update of the covariance matrix $\boldsymbol{C}^{(g)}$ via the evolution path $\boldsymbol{p}_{\mathrm{c}}^{(g+1)}$ reads [14]

$$\boldsymbol{C}^{(g+1)} = (1 - c_1)\boldsymbol{C}^{(g)} + c_1 \boldsymbol{p}_{\mathrm{c}}^{(g+1)} {\boldsymbol{p}_{\mathrm{c}}^{(g+1)}}^{\mathrm{T}} \quad . \tag{26}$$

[15] In the final algorithm (22) is still slightly modified, compare (42).

An empirically validated choice for the learning rate in (26) is $c_1 \approx 2/n^2$. For $c_\mathrm{c} = 1$ and $\mu = 1$, Equations (26), (20), and (14) are identical.

Using the evolution path for the update of $\boldsymbol{C}$ is a significant improvement of (14) for small μ_eff, because correlations between consecutive steps are exploited. The leading signs of steps, and the dependencies between consecutive steps play a significant role for the resulting evolution path $\boldsymbol{p}_\mathrm{c}^{(g+1)}$. For $c_\mathrm{c} \approx 3/n$ the number of function evaluations needed to adapt a nearly optimal covariance matrix on cigar-like objective functions becomes $\mathcal{O}(n)$.

As a last step, we combine (14) and (26).

3.4 Combining Rank-μ-Update and Cumulation

The final CMA update of the covariance matrix combines (14) and (26).

$$\boxed{\begin{aligned} \boldsymbol{C}^{(g+1)} &= (1 - c_1 - c_\mu)\,\boldsymbol{C}^{(g)} \\ &\quad + c_1\,\underbrace{\boldsymbol{p}_\mathrm{c}^{(g+1)}\boldsymbol{p}_\mathrm{c}^{(g+1)^\mathrm{T}}}_{\text{rank-one update}} + c_\mu \underbrace{\sum_{i=1}^{\mu} w_i\,\boldsymbol{y}_{i:\lambda}^{(g+1)}\left(\boldsymbol{y}_{i:\lambda}^{(g+1)}\right)^\mathrm{T}}_{\text{rank-}\mu\text{ update}} \qquad (27)\end{aligned}}$$

where

$c_1 \approx 2/n^2$.

$c_\mu \approx \min(\mu_\mathrm{eff}/n^2, 1 - c_1)$.

$\boldsymbol{y}_{i:\lambda}^{(g+1)} = (\boldsymbol{x}_{i:\lambda}^{(g+1)} - \boldsymbol{m}^{(g)})/\sigma^{(g)}$.

Equation (27) reduces to (14) for $c_1 = 0$ and to (26) for $c_\mu = 0$. The equation combines the advantages of (14) and (26). On the one hand, the information within the population of one generation is used efficiently by the rank-μ update. On the other hand, information of correlations between generations is exploited by using the evolution path for the rank-one update. The former is important in large populations, the latter is in particular important in small populations.

4 Step-Size Control

The covariance matrix adaptation, introduced in the last section, does not explicitly control the "overall scale" of the distribution, the step-size. The covariance matrix adaptation increases the scale only in one direction for each selected step, and it decreases the scale only implicitly by fading out old information via the factor $1 - c_1 - c_\mu$. Less informally, we find two specific reasons to introduce a step-size control in addition to the adaptation rule (27) for $\boldsymbol{C}^{(g)}$.

1. The optimal overall step length cannot be well approximated by (27), in particular if μ_eff is chosen larger than one.

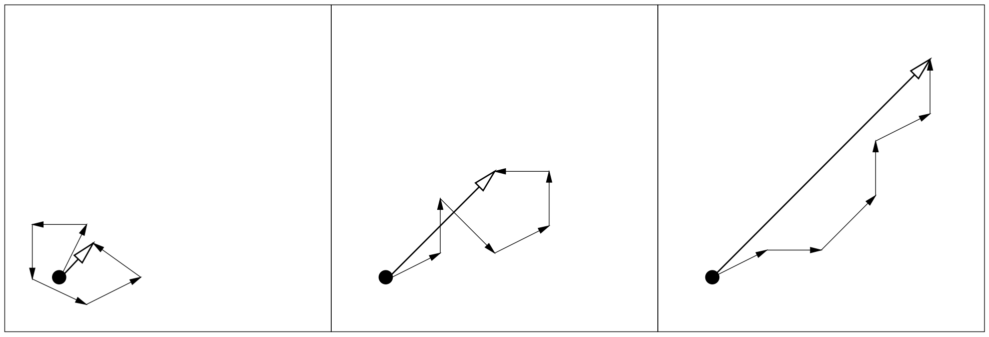


Figure 5: Three evolution paths of respectively six steps from different selection situations (idealized). The lengths of the *single* steps are all comparable. The length of the evolution paths (sum of steps) is remarkably different and is exploited for step-size control

> For example, on $f_{\mathrm{sphere}}(\boldsymbol{x}) = \sum_{i=1}^{n} x_i^2$, the optimal step-size σ equals approximately $\mu\sqrt{f_{\mathrm{sphere}}(\boldsymbol{x})}/n$, given $\boldsymbol{C}^{(g)} \approx \mathbf{I}$ and $\mu_{\mathrm{eff}} = \mu \ll n$ [3, 20]. This dependency on μ cannot be realized by (14), and is also not well approximated by (27).

2. The largest reliable learning rate for the covariance matrix update in (27) is too slow to achieve competitive change rates for the overall step length.

> To achieve optimal performance on f_{sphere} with an Evolution Strategy, the overall step length must decrease by a factor of approximately $\exp(0.202) \approx 1.22$ within n function evaluations, as can be derived from progress formulas [3, p. 229]. That is, the time horizon for the step length change must be proportional to n or shorter. From the learning rates c_1 and c_μ in (27) follows that the adaptation is too slow to perform competitive on f_{sphere} whenever $\mu_{\mathrm{eff}} \ll n$. This can be validated by simulations even for moderate dimensions, say, $n \geq 10$ and small μ_{eff}, say, $\mu_{\mathrm{eff}} \leq 1 + \ln n$.

To control the step-size $\sigma^{(g)}$ we utilize an evolution path, *i.e.* a sum of successive steps (see page 16). The method can be applied independently of the covariance matrix update and is denoted as *cumulative path length control*, cumulative step-size control, or *cumulative step length adaptation (CSA)*. The length of an evolution path is exploited, based on the following reasoning (compare also Fig. 5).

- Whenever the evolution path is short, single steps cancel each other out (Fig. 5, left). Loosely speaking, they are anti-correlated. If steps annihilate each other, the step-size should be decreased.

- Whenever the evolution path is long, the single steps are pointing to similar directions (Fig. 5, right). Loosely speaking, they are correlated. Because the steps are similar, the same distance can be covered by fewer but longer steps into the same directions. In the limit case, where consecutive steps have identical direction, they can be replaced by an enlarged single step. Consequently, the step-size should be increased.

- Subsuming, in the desired situation the steps are (approximately) perpendicular in expectation and therefore uncorrelated (Fig. 5, middle).

To decide whether the evolution path is "long" or "short", we compare the length of the path with its *expected length under random selection*.[16] Under random selection consecutive steps are independent and therefore uncorrelated (we just realized that "uncorrelated" steps are the desired situation). If selection biases the evolution path to be longer then expected, σ is increased, and, vice versa, if selection biases the evolution path to be shorter than expected, σ is decreased. In the ideal situation, selection does not bias the length of the evolution path and the length equals its expected length under random selection.

In practice, to construct the evolution path, $\boldsymbol{p}_\sigma$, the same techniques as in (22) are applied. In contrast to (22), a *conjugate* evolution path is constructed, because the expected length of the evolution path $\boldsymbol{p}_\mathrm{c}$ from (22) depends on its direction (compare (23)). Initialized with $\boldsymbol{p}_\sigma^{(0)} = \boldsymbol{0}$, the conjugate evolution path reads

$$\boxed{\boldsymbol{p}_\sigma^{(g+1)} = (1-c_\sigma)\boldsymbol{p}_\sigma^{(g)} + \sqrt{c_\sigma(2-c_\sigma)\mu_\mathrm{eff}}\, {\boldsymbol{C}^{(g)}}^{-\frac{1}{2}}\, \frac{\boldsymbol{m}^{(g+1)} - \boldsymbol{m}^{(g)}}{\sigma^{(g)}} \qquad (28)}$$

where

$\boldsymbol{p}_\sigma^{(g)} \in \mathbb{R}^n$ is the conjugate evolution path at generation g.

$c_\sigma < 1$. Again, $1/c_\sigma$ is the backward time horizon of the evolution path (compare (18)). For small μ_eff, a time horizon between $\sqrt{n}$ and n is reasonable.

$\sqrt{c_\sigma(2-c_\sigma)\mu_\mathrm{eff}}$ is a normalization constant, see (22).

${\boldsymbol{C}^{(g)}}^{-\frac{1}{2}} \overset{\mathrm{def}}{=} \boldsymbol{B}^{(g)}{\boldsymbol{D}^{(g)}}^{-1}{\boldsymbol{B}^{(g)}}^\mathrm{T}$, where $\boldsymbol{C}^{(g)} = \boldsymbol{B}^{(g)}\left(\boldsymbol{D}^{(g)}\right)^2{\boldsymbol{B}^{(g)}}^\mathrm{T}$ is an eigendecomposition of $\boldsymbol{C}^{(g)}$, where $\boldsymbol{B}^{(g)}$ is an orthonormal basis of eigenvectors, and the diagonal elements of the diagonal matrix $\boldsymbol{D}^{(g)}$ are square roots of the corresponding positive eigenvalues (cf. Sect. 0.1).

For $\boldsymbol{C}^{(g)} = \mathbf{I}$, we have ${\boldsymbol{C}^{(g)}}^{-\frac{1}{2}} = \mathbf{I}$ and (28) replicates (22). The transformation ${\boldsymbol{C}^{(g)}}^{-\frac{1}{2}}$ re-scales the step $\boldsymbol{m}^{(g+1)} - \boldsymbol{m}^{(g)}$ within the coordinate system given by $\boldsymbol{B}^{(g)}$.

The single factors of the transformation ${\boldsymbol{C}^{(g)}}^{-\frac{1}{2}} = \boldsymbol{B}^{(g)}{\boldsymbol{D}^{(g)}}^{-1}{\boldsymbol{B}^{(g)}}^\mathrm{T}$ can be explained as follows (from right to left):

${\boldsymbol{B}^{(g)}}^\mathrm{T}$ rotates the space such that the columns of $\boldsymbol{B}^{(g)}$, *i.e.* the principal axes of the distribution $\mathcal{N}(\boldsymbol{0}, \boldsymbol{C}^{(g)})$, rotate into the coordinate axes. Elements of the resulting vector relate to projections onto the corresponding eigenvectors.

${\boldsymbol{D}^{(g)}}^{-1}$ applies a (re-)scaling such that all axes become equally sized.

$\boldsymbol{B}^{(g)}$ rotates the result back into the original coordinate system. This last transformation ensures that the principal axes of the distribution are not rotated by the overall transformation and directions of consecutive steps are comparable.

[16] Random selection means that the index $i:\lambda$ (compare (6)) is independent of the value of $\boldsymbol{x}_{i:\lambda}^{(g+1)}$ for all $i = 1,\ldots,\lambda$, e.g. $i:\lambda = i$.

Consequently, the transformation ${\boldsymbol{C}^{(g)}}^{-\frac{1}{2}}$ makes the expected length of $\boldsymbol{p}_\sigma^{(g+1)}$ independent of its direction, and for any sequence of realized covariance matrices $\boldsymbol{C}_{g=0,1,2,\dots}^{(g)}$ we have under random selection $\boldsymbol{p}_\sigma^{(g+1)} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, given $\boldsymbol{p}_\sigma^{(0)} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ [7].

To update $\sigma^{(g)}$, we "compare" $\|\boldsymbol{p}_\sigma^{(g+1)}\|$ with its expected length $\mathsf{E}\|\mathcal{N}(\mathbf{0}, \mathbf{I})\|$, that is

$$\ln \sigma^{(g+1)} = \ln \sigma^{(g)} + \frac{c_\sigma}{d_\sigma \mathsf{E}\|\mathcal{N}(\mathbf{0}, \mathbf{I})\|} \left(\|\boldsymbol{p}_\sigma^{(g+1)}\| - \mathsf{E}\|\mathcal{N}(\mathbf{0}, \mathbf{I})\| \right) \quad , \tag{29}$$

where

$d_\sigma \approx 1$, damping parameter, scales the change magnitude of $\ln \sigma^{(g)}$. The factor $c_\sigma / d_\sigma / \mathsf{E}\|\mathcal{N}(\mathbf{0}, \mathbf{I})\|$ is based on in-depth investigations of the algorithm [7].

$\mathsf{E}\|\mathcal{N}(\mathbf{0}, \mathbf{I})\| = \sqrt{2}\,\Gamma(\frac{n+1}{2})/\Gamma(\frac{n}{2}) \approx \sqrt{n} + \mathcal{O}(1/n)$, expectation of the Euclidean norm of a $\mathcal{N}(\mathbf{0}, \mathbf{I})$ distributed random vector.

For $\|\boldsymbol{p}_\sigma^{(g+1)}\| = \mathsf{E}\|\mathcal{N}(\mathbf{0}, \mathbf{I})\|$ the second summand in (29) is zero, and $\sigma^{(g)}$ is unchanged, while $\sigma^{(g)}$ is increased for $\|\boldsymbol{p}_\sigma^{(g+1)}\| > \mathsf{E}\|\mathcal{N}(\mathbf{0}, \mathbf{I})\|$, and $\sigma^{(g)}$ is decreased for $\|\boldsymbol{p}_\sigma^{(g+1)}\| < \mathsf{E}\|\mathcal{N}(\mathbf{0}, \mathbf{I})\|$.

Alternatively, we might use the squared norm $\|\boldsymbol{p}_\sigma^{(g+1)}\|^2$ in (29) and compare with its expected value n [2]. In this case (29) would read

$$\begin{aligned} \ln \sigma^{(g+1)} &= \ln \sigma^{(g)} + \frac{c_\sigma}{d_\sigma 2n} \left(\|\boldsymbol{p}_\sigma^{(g+1)}\|^2 - n \right) \\ &= \ln \sigma^{(g)} + \frac{c_\sigma}{2 d_\sigma} \left(\frac{\|\boldsymbol{p}_\sigma^{(g+1)}\|^2}{n} - 1 \right) \quad . \end{aligned} \tag{30}$$

This update will presumable lead to faster step-size increments and slower step-size decrements.

The step-size change is unbiased on the log scale, because $\mathsf{E}\left[\ln \sigma^{(g+1)} \middle| \sigma^{(g)} \right] = \ln \sigma^{(g)}$ for $\boldsymbol{p}_\sigma^{(g+1)} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. The role of unbiasedness is discussed in Sect. 5. Equations (28) and (29) cause successive steps of the distribution mean $\boldsymbol{m}^{(g)}$ to be approximately ${\boldsymbol{C}^{(g)}}^{-1}$-conjugate.

In order to show that successive steps are approximately ${\boldsymbol{C}^{(g)}}^{-1}$-conjugate first we remark that (28) and (29) adapt σ such that the length of $\boldsymbol{p}_\sigma^{(g+1)}$ equals approximately $\mathsf{E}\|\mathcal{N}(\mathbf{0}, \mathbf{I})\|$. Starting from $(\mathsf{E}\|\mathcal{N}(\mathbf{0}, \mathbf{I})\|)^2 \approx \|\boldsymbol{p}_\sigma^{(g+1)}\|^2 = {\boldsymbol{p}_\sigma^{(g+1)}}^{\mathrm{T}} \boldsymbol{p}_\sigma^{(g+1)} =$ $\mathrm{RHS}^{\mathrm{T}}\mathrm{RHS}$ of (28) and assuming that the expected squared *length* of ${\boldsymbol{C}^{(g)}}^{-\frac{1}{2}}(\boldsymbol{m}^{(g+1)} - \boldsymbol{m}^{(g)})$ is unchanged by selection (unlike its direction) we get

$${\boldsymbol{p}_\sigma^{(g)}}^{\mathrm{T}} {\boldsymbol{C}^{(g)}}^{-\frac{1}{2}} (\boldsymbol{m}^{(g+1)} - \boldsymbol{m}^{(g)}) \approx 0 \quad , \tag{31}$$

and

$$\left({\boldsymbol{C}^{(g)}}^{\frac{1}{2}} \boldsymbol{p}_\sigma^{(g)} \right)^{\mathrm{T}} {\boldsymbol{C}^{(g)}}^{-1} \left(\boldsymbol{m}^{(g+1)} - \boldsymbol{m}^{(g)} \right) \approx 0 \quad . \tag{32}$$

Given $1/(c_1 + c_\mu) \gg 1$ and (31) we assume also ${\boldsymbol{p}_\sigma^{(g-1)}}^{\mathrm{T}} {\boldsymbol{C}^{(g)}}^{-\frac{1}{2}} (\boldsymbol{m}^{(g+1)} - \boldsymbol{m}^{(g)}) \approx 0$ and derive

$$\left(\boldsymbol{m}^{(g)} - \boldsymbol{m}^{(g-1)}\right)^{\mathrm{T}} {\boldsymbol{C}^{(g)}}^{-1} \left(\boldsymbol{m}^{(g+1)} - \boldsymbol{m}^{(g)}\right) \approx 0 \ . \tag{33}$$

That is, the steps taken by the distribution mean become approximately ${\boldsymbol{C}^{(g)}}^{-1}$-conjugate.

Because $\sigma^{(g)} > 0$, (29) is equivalent to

$$\boxed{\sigma^{(g+1)} = \sigma^{(g)} \exp\left(\frac{c_\sigma}{d_\sigma}\left(\frac{\|\boldsymbol{p}_\sigma^{(g+1)}\|}{\mathsf{E}\|\mathcal{N}(\mathbf{0},\mathbf{I})\|} - 1\right)\right)} \tag{34}$$

The length of the evolution path is an intuitive and empirically well validated goodness measure for the overall step length. For $\mu_{\mathrm{eff}} > 1$ it is the best measure to our knowledge. Nevertheless, it fails to adapt nearly optimal step-sizes on very noisy objective functions [4].

5 Discussion

The CMA-ES is an attractive option for non-linear optimization, if "classical" search methods, e.g. quasi-Newton methods (BFGS) and/or conjugate gradient methods, fail due to a non-convex or rugged search landscape (e.g. sharp bends, discontinuities, outliers, noise, and local optima). Learning the covariance matrix in the CMA-ES is analogous to learning the inverse Hessian matrix in a quasi-Newton method. In the end, any convex-quadratic (ellipsoid) objective function is transformed into the spherical function f_{sphere}. This can improve the performance on ill-conditioned and/or non-separable problems by orders of magnitude.

The CMA-ES overcomes typical problems that are often associated with evolutionary algorithms.

1. Poor performance on badly scaled and/or highly non-separable objective functions. Equation (27) adapts the search distribution to badly scaled and non-separable problems.

2. The inherent need to use large population sizes. A typical, however intricate to diagnose reason for the failure of population based search algorithms is the degeneration of the population into a subspace. This is usually prevented by non-adaptive components in the algorithm and/or by a large population size (considerably larger than the problem dimension). In the CMA-ES, the population size can be freely chosen, because the learning rates c_1 and c_μ in (27) prevent the degeneration even for small population sizes, e.g. $\lambda = 9$. Small population sizes usually lead to faster convergence, large population sizes help to avoid local optima.

3. Premature convergence of the population. Step-size control in (34) prevents the population to converge prematurely. It does not prevent the search to end up in a local optimum.

Therefore, the CMA-ES is highly competitive on a considerable number of test functions [7, 11, 13, 15, 16] and was successfully applied to many real world problems.[17] Finally we discuss a few **basic design principles** that were applied in the previous sections.

Change rates We refer to a change rate as the expected parameter change *per sampled search point*, given a certain selection situation. To achieve competitive performance on a wide range of objective functions, the possible change rates of the adaptive parameters need to be adjusted carefully. The CMA-ES separately controls change rates for the mean value of the distribution, $\boldsymbol{m}$, the covariance matrix, $\boldsymbol{C}$, and the step-size, σ.

- The change rate for the mean value $\boldsymbol{m}$, given a fixed sample distribution, is determined by the parent number and the recombination weights. The larger μ_{eff}, the smaller is the possible change rate of $\boldsymbol{m}$. Similar holds for most evolutionary algorithms.
- The change rate of the covariance matrix $\boldsymbol{C}$ is explicitly controlled by the learning rates c_1 and c_μ and therefore detached from parent number and population size. The learning rate reflects the model complexity. In evolutionary algorithms, the explicit control of change rates of covariances, independent from the population size, is a rare feature.
- The change rate of the step-size σ is explicitly controlled by the damping parameter d_σ and is in particular independent from the change rate of $\boldsymbol{C}$. The time constant $1/c_\sigma \leq n$ ensures a sufficiently fast change of the overall step length in particular with small population sizes.

Invariance Invariance properties of a search algorithm denote identical behavior on a set, or a class of objective functions. Invariance is an important property of the CMA-ES.[18] Translation invariance should be taken for granted in continuous domain optimization. Translation invariance means that the search behavior on the function $\boldsymbol{x} \mapsto f(\boldsymbol{x} + \boldsymbol{a})$, $\boldsymbol{x}^{(0)} = \boldsymbol{b} - \boldsymbol{a}$, is independent of $\boldsymbol{a} \in \mathbb{R}^n$. Further invariances, e.g. to certain linear transformations of the search space, are highly desirable: they imply uniform performance on classes of functions and therefore allow for generalization of empirical results. The CMA-ES exhibits the following invariances.

- Invariance to order preserving (*i.e.* strictly monotonic) transformations of the objective function value. The algorithm only depends on *the ranking* of function values.
- Invariance to angle preserving (rigid) transformations of the search space (rotation, reflection, and translation) if the initial search point is transformed accordingly.
- Scale invariance if the initial scaling, e.g. $\sigma^{(0)}$, and the initial search point, $\boldsymbol{m}^{(0)}$, are chosen accordingly.
- Invariance to a scaling of variables if the initial diagonal covariance matrix $\boldsymbol{C}^{(0)}$, and the initial search point, $\boldsymbol{m}^{(0)}$, are chosen accordingly.

[17] For a list of published applications see http://www.lri.fr/~hansen/cmaapplications.pdf

[18] Special acknowledgments to Iván Santibáñez-Koref for pointing this out to me.

- Invariance to any invertible linear transformation of the search space, $\boldsymbol{A}$, if the initial covariance matrix $\boldsymbol{C}^{(0)} = \boldsymbol{A}^{-1}\left(\boldsymbol{A}^{-1}\right)^{\mathrm{T}}$, and if the initial search point, $\boldsymbol{m}^{(0)}$, are transformed accordingly.

Invariance should be a fundamental design criterion for any search algorithm. Together with the ability to efficiently adapt the invariance governing parameters, invariance is a key to competitive performance.

Stationarity An important design criterion for a stochastic search procedure is *unbiasedness* of variations of object and strategy parameters [5, 16]. Consider random selection, e.g. the objective function $f(\boldsymbol{x}) = \texttt{rand}$ to be independent of $\boldsymbol{x}$. The population mean is unbiased if its expected value remains unchanged in the next generation, that is $\mathsf{E}\left[\boldsymbol{m}^{(g+1)} \,\middle|\, \boldsymbol{m}^{(g)}\right] = \boldsymbol{m}^{(g)}$. For the population mean, stationarity under random selection is a rather intuitive concept. In the CMA-ES, stationarity is respected for all parameters in the basic equation (5). The distribution mean $\boldsymbol{m}$, the covariance matrix $\boldsymbol{C}$, and $\ln\sigma$ are unbiased. Unbiasedness of $\ln\sigma$ does not imply that σ is unbiased. Actually, under random selection, $\mathsf{E}\left[\sigma^{(g+1)} \,\middle|\, \sigma^{(g)}\right] > \sigma^{(g)}$, compare (29).[19]

For distribution variances (or step-sizes) a bias toward increase or decrease entails the danger of divergence or premature convergence, respectively, whenever the selection pressure is low. Nevertheless, on noisy problems a properly controlled bias towards increase might be appropriate even on the log scale.

Acknowledgments

The author wishes to gratefully thank Anne Auger, Christian Igel, Stefan Kern, and Fabrice Marchal for the many valuable comments on the manuscript.

References

[1] Auger A, Hansen N. A restart CMA evolution strategy with increasing population size. In *Proceedings of the IEEE Congress on Evolutionary Computation*, 2005.

[2] Arnold DV, Beyer HG. Performance analysis of evolutionary optimization with cumulative step length adaptation. *IEEE Transactions on Automatic Control*, 49(4):617–622, 2004.

[3] Beyer HG. *The Theory of Evolution Strategies*. Springer, Berlin, 2001.

[4] Beyer HG, Arnold DV. Qualms regarding the optimality of cumulative path length control in CSA/CMA-evolution strategies. *Evolutionary Computation*, 11(1):19–28, 2003.

[5] Beyer HG, Deb K. On self-adaptive features in real-parameter evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 5(3):250–270, 2001.

[19] Alternatively, if (34) would be designed to be unbiased for $\sigma^{(g+1)}$, this would imply that $\mathsf{E}\left[\ln\sigma^{(g+1)} \,\middle|\, \sigma^{(g)}\right] < \ln\sigma^{(g)}$, to our opinion a less desirable variant.

[6] G. Collange, N. Delattre, N. Hansen, I. Quinquis, and M. Schoenauer. Multidisciplinary optimisation in the design of future space launchers. In P. Breitkopf and R. F. Coelho, editors, *Multidisciplinary Design Optimization in Computational Mechanics*, chapter 12, pages 487–496. Wiley, 2010.

[7] Hansen N. *Verallgemeinerte individuelle Schrittweitenregelung in der Evolutionsstrategie.* Mensch und Buch Verlag, Berlin, 1998.

[8] Hansen N. Invariance, self-adaptation and correlated mutations in evolution strategies. In Schoenauer M, Deb K, Rudolph G, Yao X, Lutton E, Merelo JJ, Schwefel HP, editors, *Parallel Problem Solving from Nature - PPSN VI*, pages 355–364. Springer, 2000.

[9] Hansen N. The CMA evolution strategy: a comparing review. In Lozano JA, Larranaga P, Inza I, and Bengoetxea E, editors, *Towards a new evolutionary computation. Advances on estimation of distribution algorithms*, pages 75–102. Springer, 2006.

[10] Hansen N. Benchmarking a BI-Population CMA-ES on the BBOB-2009 Function Testbed. In the workshop *Proceedings of the GECCO Genetic and Evolutionary Computation Conference*, pages 2389–2395. ACM, 2009.

[11] Hansen N, Kern S. Evaluating the CMA evolution strategy on multimodal test functions. In Xin Yao et al., editors, *Parallel Problem Solving from Nature - PPSN VIII*, pages 282–291. Springer, 2004.

[12] Hansen N, Niederberger SPN, Guzzella L, Koumoutsakos P. A method for handling uncertainty in evolutionary optimization with an application to feedback control of combustion. *IEEE Transactions on Evolutionary Computation*, 2009.

[13] Hansen N, Müller SD, Koumoutsakos P. Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES). *Evolutionary Computation*, 11(1):1–18, 2003.

[14] Hansen N, Ostermeier A. Adapting arbitrary normal mutation distributions in evolution strategies: The covariance matrix adaptation. In *Proceedings of the 1996 IEEE Conference on Evolutionary Computation (ICEC '96)*, pages 312–317, 1996.

[15] Hansen N, Ostermeier A. Convergence properties of evolution strategies with the derandomized covariance matrix adaptation: The $(\mu/\mu_I, \lambda)$-CMA-ES. In *Proceedings of the 5^{th} European Congresson Intelligent Techniques and Soft Computing*, pages 650–654, 1997.

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

[17] Kern S, Müller SD, Hansen N, Büche D, Ocenasek J, Koumoutsakos P. Learning probability distributions in continuous evolutionary algorithms – a comparative review. *Natural Computing*, 3:77–112, 2004.

[18] Larrañaga P. A review on estimation of distribution algorithms. In P. Larrañaga and J. A. Lozano, editors, *Estimation of Distribution Algorithms*, pages 80–90. Kluwer Academic Publishers, 2002.

[19] Larrañaga P, Lozano JA, Bengoetxea E. Estimation of distribution algorithms based on multivariate normal and Gaussian networks. Technical report, Dept. of Computer Science and Artificial Intelligence, University of the Basque Country, 2001. KZAA-IK-1-01.

[20] Rechenberg I. *Evolutionsstrategie '94*. Frommann-Holzboog, Stuttgart, Germany, 1994.

[21] Rubenstein RY, Kroese DP. The Cross-Entropy Method: a unified approach to combinatorial optimization, Monte-Carlo simulation, and machine learning. Springer, 2004.

A Algorithm Summary: The $(\mu/\mu_{\mathrm{W}}, \lambda)$-CMA-ES

Figure 6 outlines the complete algorithm, summarizing (5), (6), (22), (27), (28), and (34). Used symbols are:

$\boldsymbol{y}_k \sim \mathcal{N}(\mathbf{0}, \boldsymbol{C})$, for $k = 1, \dots, \lambda$, are realizations from a multivariate normal distribution with zero mean and covariance matrix $\boldsymbol{C}$.

$\boldsymbol{B}, \boldsymbol{D}$ result from an eigendecomposition of the covariance matrix $\boldsymbol{C}$ with $\boldsymbol{C} = \boldsymbol{B}\boldsymbol{D}^2\boldsymbol{B}^{\mathrm{T}} = \boldsymbol{B}\boldsymbol{D}\boldsymbol{D}\boldsymbol{B}^{\mathrm{T}}$ (cf. Sect. 0.1). Columns of $\boldsymbol{B}$ are an orthonormal basis of eigenvectors. Diagonal elements of the diagonal matrix $\boldsymbol{D}$ are square roots of the corresponding positive eigenvalues. While (36) can certainly be implemented using a Cholesky decomposition of $\boldsymbol{C}$ the eigendecomposition is needed to compute (40) correctly.

$\boldsymbol{x}_k \in \mathbb{R}^n$, for $k = 1, \dots, \lambda$. Sample of λ search points.

$\langle \boldsymbol{y} \rangle_{\mathrm{w}} = \sum_{i=1}^{\mu} w_i\, \boldsymbol{y}_{i:\lambda}$, step of the distribution mean disregarding step-size σ.

$\boldsymbol{y}_{i:\lambda} = (\boldsymbol{x}_{i:\lambda} - \boldsymbol{m})/\sigma$, see $\boldsymbol{x}_{i:\lambda}$ below.

$\boldsymbol{x}_{i:\lambda} \in \mathbb{R}^n$, i-th best point out of $\boldsymbol{x}_1, \dots, \boldsymbol{x}_\lambda$ from (37). The index $i:\lambda$ denotes the index of the i-th ranked point, that is $f(\boldsymbol{x}_{1:\lambda}) \le f(\boldsymbol{x}_{2:\lambda}) \le \dots \le f(\boldsymbol{x}_{\lambda:\lambda})$.

$\mu_{\mathrm{eff}} = \left(\sum_{i=1}^{\mu} w_i^2\right)^{-1}$ is the variance effective selection mass, see (8). It holds $1 \le \mu_{\mathrm{eff}} \le \mu$.

$\boldsymbol{C}^{-\frac{1}{2}} \overset{\mathrm{def}}{=} \boldsymbol{B}\boldsymbol{D}^{-1}\boldsymbol{B}^{\mathrm{T}}$, see $\boldsymbol{B}, \boldsymbol{D}$ above. The matrix $\boldsymbol{D}$ can be inverted by inverting its diagonal elements. From the definitions we find that $\boldsymbol{C}^{-\frac{1}{2}} \langle \boldsymbol{y} \rangle_{\mathrm{w}} = \boldsymbol{B} \langle \boldsymbol{z} \rangle_{\mathrm{w}}$ with $\langle \boldsymbol{z} \rangle_{\mathrm{w}} = \sum_{i=1}^{\mu} w_i\, \boldsymbol{z}_{i:\lambda}$.

$\mathsf{E}\|\mathcal{N}(\mathbf{0}, \mathbf{I})\| = \sqrt{2}\,\Gamma(\frac{n+1}{2})/\Gamma(\frac{n}{2}) \approx \sqrt{n}\left(1 - \frac{1}{4n} + \frac{1}{21n^2}\right)$.

$h_\sigma = \begin{cases} 1 & \text{if } \frac{\|\boldsymbol{p}_\sigma\|}{\sqrt{1-(1-c_\sigma)^{2(g+1)}}} < (1.4 + \frac{2}{n+1})\mathsf{E}\|\mathcal{N}(\mathbf{0}, \mathbf{I})\| \\ 0 & \text{otherwise} \end{cases}$, where g is the generation number. The Heaviside function h_σ stalls the update of $\boldsymbol{p}_{\mathrm{c}}$ in (42) if $\|\boldsymbol{p}_\sigma\|$ is large. This prevents a too fast increase of axes of $\boldsymbol{C}$ in a linear surrounding, *i.e.* when the step-size is far too small. This is useful when the initial step-size is chosen far too small or when the objective function changes in time.

$\delta(h_\sigma) = (1 - h_\sigma)c_{\mathrm{c}}(2 - c_{\mathrm{c}}) \le 1$ is of minor relevance. In the (unusual) case of $h_\sigma = 0$, it substitutes for the second summand from (42) in (43).

Set parameters

Set parameters λ, μ, $w_{i=1\ldots\mu}$, c_σ, d_σ, c_c, c_1, and c_μ to their default values according to Table 1.

Initialization

Set evolution paths $\boldsymbol{p}_\sigma = \mathbf{0}$, $\boldsymbol{p}_\mathrm{c} = \mathbf{0}$, covariance matrix $\boldsymbol{C} = \mathbf{I}$, and $g = 0$.

Choose distribution mean $\boldsymbol{m} \in \mathbb{R}^n$ and step-size $\sigma \in \mathbb{R}_+$ problem dependent.[1]

Until termination criterion met, $g \leftarrow g + 1$

Sample new population of search points, for $k = 1, \ldots, \lambda$

$$\boldsymbol{z}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{I}) \tag{35}$$

$$\boldsymbol{y}_k = \boldsymbol{B}\boldsymbol{D}\boldsymbol{z}_k \quad \sim \mathcal{N}(\mathbf{0}, \boldsymbol{C}) \tag{36}$$

$$\boldsymbol{x}_k = \boldsymbol{m} + \sigma \boldsymbol{y}_k \quad \sim \mathcal{N}\big(\boldsymbol{m}, \sigma^2\boldsymbol{C}\big) \tag{37}$$

Selection and recombination

$$\langle \boldsymbol{y} \rangle_\mathrm{w} = \sum_{i=1}^{\mu} w_i\, \boldsymbol{y}_{i:\lambda} \quad \text{where} \sum_{i=1}^{\mu} w_i = 1,\ w_i > 0 \tag{38}$$

$$\boldsymbol{m} \leftarrow \boldsymbol{m} + \sigma \langle \boldsymbol{y} \rangle_\mathrm{w} = \sum_{i=1}^{\mu} w_i\, \boldsymbol{x}_{i:\lambda} \tag{39}$$

Step-size control

$$\boldsymbol{p}_\sigma \leftarrow (1 - c_\sigma)\boldsymbol{p}_\sigma + \sqrt{c_\sigma(2 - c_\sigma)\mu_\mathrm{eff}}\, \boldsymbol{C}^{-\frac{1}{2}} \langle \boldsymbol{y} \rangle_\mathrm{w} \tag{40}$$

$$\sigma \leftarrow \sigma \times \exp\left(\frac{c_\sigma}{d_\sigma}\left(\frac{\|\boldsymbol{p}_\sigma\|}{\mathsf{E}\|\mathcal{N}(\mathbf{0}, \mathbf{I})\|} - 1\right)\right) \tag{41}$$

Covariance matrix adaptation

$$\boldsymbol{p}_\mathrm{c} \leftarrow (1 - c_\mathrm{c})\boldsymbol{p}_\mathrm{c} + h_\sigma \sqrt{c_\mathrm{c}(2 - c_\mathrm{c})\mu_\mathrm{eff}}\, \langle \boldsymbol{y} \rangle_\mathrm{w} \tag{42}$$

$$\boldsymbol{C} \leftarrow (1 - c_1 - c_\mu)\, \boldsymbol{C} + c_1 \left(\boldsymbol{p}_\mathrm{c}\boldsymbol{p}_\mathrm{c}^\mathrm{T} + \delta(h_\sigma)\boldsymbol{C}\right) + c_\mu \sum_{i=1}^{\mu} w_i\, \boldsymbol{y}_{i:\lambda}\boldsymbol{y}_{i:\lambda}^\mathrm{T} \tag{43}$$

[1]The optimum should presumably be within the initial cube $\boldsymbol{m} \pm 3\sigma(1, \ldots, 1)^\mathrm{T}$. If the optimum is expected to be in the initial search interval $[a, b]^n$ we may choose the initial search point, $\boldsymbol{m}$, uniformly randomly in $[a, b]^n$, and $\sigma = 0.3(b - a)$. Different search intervals Δs_i for different variables can be reflected by a different initialization of $\boldsymbol{C}$, in that the diagonal elements of $\boldsymbol{C}$ obey $c_{ii} = (\Delta s_i)^2$. Remark that the Δs_i should not disagree by several orders of magnitude. Otherwise a scaling of the variables should be applied.

Figure 6: The $(\mu/\mu_\mathrm{W}, \lambda)$-CMA Evolution Strategy. Symbols: see text

Table 1: Default Strategy Parameters, where $\mu_{\text{eff}} = \frac{1}{\sum_{i=1}^{\mu} w_i^2} \geq 1$ and $\sum_{i=1}^{\mu} w_i = 1$, taken from [10]

Selection and Recombination:

$$\lambda = 4 + \lfloor 3 \ln n \rfloor, \quad \mu = \lfloor \mu' \rfloor, \quad \mu' = \frac{\lambda}{2} \tag{44}$$

$$w_i = \frac{w'_i}{\sum_{j=1}^{\mu} w'_j}, \quad w'_i = \ln(\mu' + 0.5) - \ln i \quad \text{for } i = 1, \ldots, \mu, \tag{45}$$

Step-size control:

$$c_\sigma = \frac{\mu_{\text{eff}} + 2}{n + \mu_{\text{eff}} + 5}, \quad d_\sigma = 1 + 2 \max\left(0, \sqrt{\frac{\mu_{\text{eff}} - 1}{n + 1}} - 1\right) + c_\sigma \tag{46}$$

Covariance matrix adaptation:

$$c_{\text{c}} = \frac{4 + \mu_{\text{eff}}/n}{n + 4 + 2\mu_{\text{eff}}/n} \tag{47}$$

$$c_1 = \frac{2}{(n + 1.3)^2 + \mu_{\text{eff}}} \tag{48}$$

$$c_\mu = \min\left(1 - c_1, \alpha_\mu \frac{\mu_{\text{eff}} - 2 + 1/\mu_{\text{eff}}}{(n + 2)^2 + \alpha_\mu \mu_{\text{eff}}/2}\right) \quad \text{with } \alpha_\mu = 2 \tag{49}$$

Default Parameters The (external) strategy parameters are λ, μ, $w_{i=1\ldots\mu}$, c_σ, d_σ, c_{c}, c_1, and c_μ. Default strategy parameters values are given in Table 1. An in-depth discussion of most parameters is given in [16]. The default parameters of (46)-(49) are in particular chosen to be a robust setting and therefore, to our experience, applicable to a wide range of functions to be optimized. We do not recomment to change this setting. In contrast, the population size λ in (44) can be increased.[20] If the λ-dependent default values for μ and w_i are used, the population size λ has a significant influence on the global search performance [11]. Increasing λ usually improves the global search capability and the robustness of the CMA-ES, at the price of a reduced convergence speed. The convergence speed decrease at most linearly with λ. Independent restarts with increasing population size [1], automated or manually conducted, are an appropriate policy to perform well on most problems.

B Implementational Concerns

We discuss a few implementational concerns.

[20]Decreasing λ is not recommended. Too small values regularly have strong adverse effects on the performance.

B.1 Multivariate normal distribution

Let the vector $\boldsymbol{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ have independent, $(0,1)$-normally distributed components that can easily be sampled on a computer. To generate a random vector $\boldsymbol{y} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{C})$ for (36), we set $\boldsymbol{y} = \boldsymbol{BDz}$ (see above symbol descriptions of $\boldsymbol{B}$ and $\boldsymbol{D}$ and Sects. 0.1 and 0.2, and compare lines 52–53 and 83–84 in the source code below). Given $\boldsymbol{y}_k = \boldsymbol{BDz}_k$ and $\boldsymbol{C}^{-\frac{1}{2}} = \boldsymbol{BD}^{-1}\boldsymbol{B}^{\mathrm{T}}$ we have $\boldsymbol{C}^{-\frac{1}{2}} \langle \boldsymbol{y} \rangle_{\mathrm{w}} = \boldsymbol{B} \sum_{i=1}^{\mu} w_i \, \boldsymbol{z}_{i:\lambda}$ (compare (40) and lines 61 and 64 in the source code below).

B.2 Strategy internal numerical effort

In practice, the re-calculation of $\boldsymbol{B}$ and $\boldsymbol{D}$ needs to be done not until $\max(1, \lfloor 1/(10n(c_1 + c_\mu)) \rfloor)$ generations. For reasonable $c_1 + c_\mu$ values, this reduces the numerical effort due to the eigendecomposition from $\mathcal{O}(n^3)$ to $\mathcal{O}(n^2)$ per generated search point, that is the effort of a matrix vector multiplication.

On a Pentium 4, 2.5 GHz processor the overall strategy internal time consumption is roughly $3 \times (n+5)^2 \times 10^{-8}$ seconds per function evaluation [17].

Remark that it is not sufficient to compute a Cholesky decomposition of $\boldsymbol{C}$, because then (40) cannot be computed correctly.

B.3 Termination criteria

In general, the algorithm should be stopped whenever it becomes a waste of CPU-time to continue, and it would be better to restart (eventually with increased population size [1]) or to reconsidering the encoding and/or objective function formulation. We recommend the following termination criteria [1, 10] that are mostly related to numerical stability:

- `NoEffectAxis`: stop if adding a 0.1-standard deviation vector in any principal axis direction of $\boldsymbol{C}$ does not change $\boldsymbol{m}$.[21]
- `NoEffectCoord`: stop if adding 0.2-standard deviations in any single coordinate does not change $\boldsymbol{m}$ (*i.e.* m_i equals $m_i + 0.2\,\sigma c_{i,i}$ for any i).
- `EqualFunValues`: stop if the range of the best objective function values of the last $10 + \lceil 30n/\lambda \rceil$ generations is zero.
- `Stagnation`: we track a history of the best and the median fitness in each iteration over the last 20% but at least $120 + 30n/\lambda$ and no more than 20 000 iterations. We stop, if in both histories the median of the last (most recent) 30% values is not better than the median of the first 30%.
- `ConditionCov`: stop if the condition number of the covariance matrix exceeds 10^{14}.
- `TolXUp`: stop if $\sigma \times \max(\mathrm{diag}(\boldsymbol{D}))$ increased by more than 10^4. This usually indicates a far too small initial σ, or divergent behavior.

[21]More formally, we terminate if $\boldsymbol{m}$ equals to $\boldsymbol{m} + 0.1\,\sigma d_{ii}\boldsymbol{b}_i$, where $i = (g \bmod n) + 1$, and d_{ii}^2 and $\boldsymbol{b}_i$ are respectively the i-th eigenvalue and eigenvector of $\boldsymbol{C}$, with $\|\boldsymbol{b}_i\| = 1$.

Two other useful termination criteria should be considered problem dependent:

- `TolFun`: stop if the range of the best objective function values of the last $10+\lceil 30n/\lambda\rceil$ generations and all function values of the recent generation is below `TolFun`. Choosing `TolFun` depends on the problem, while 10^{-12} is a conservative first guess.
- `TolX`: stop if the standard deviation of the normal distribution is smaller than in all coordinates and $\sigma \boldsymbol{p}_\mathrm{c}$ is smaller than `TolX` in all components. By default we set `TolX` to 10^{-12} times the initial σ.

B.4 Flat fitness

In the case of equal function values for several individuals in the population, it is feasible to increase the step-size (see lines 92–96 in the source code below). This method can interfere with the termination criterion `TolFun`. In practice, observation of a flat fitness should be rather a termination criterion and consequently lead to a reconsideration of the objective function formulation.

B.5 Boundaries and Constraints.

The handling of boundaries and constraints is to a certain extend problem dependent. We discuss a few useful approaches.

Best solution is stricly inside If the optimal solution is *not too close to the infeasible domain* (or at least not in a "corner"), a simple and sufficient way to handle any type of boundaries and constraints are

1. set the fitness as
$$f_\mathrm{fitness}(\boldsymbol{x}) = f_\mathrm{max} + \|\boldsymbol{x} - \boldsymbol{x}_\mathrm{feasible}\| \ , \tag{50}$$
where f_max is larger than the worst fitness in the feasible population or in the feasible domain (in case of minization) and $\boldsymbol{x}_\mathrm{feasible}$ is a constant feasible point, preferably in the middle of the feasible domain.
2. re-sampling any infeasible solution $\boldsymbol{x}$ until it become feasible.

Repairement available as for example with box-constraints.

Simple repairment It is possible to simply repair infeasible individuals before the update equations are applied. This is not recommended, because the CMA-ES makes implicite assumptions on the distribution of solution points, which can be heavily violated by a repairment. The main resulting problem might be divergence or too fast convergence of the step-size (however a solution will soon be available). Note also, that this might be intricate to implement, in particular if $\boldsymbol{y}$ or $\boldsymbol{z}$ are used for implementing the update equations in the original code.

Penalization We evaluate the objective function on a repaired search point, $\boldsymbol{x}_\mathrm{repaired}$, and add a penalty depending on the distance to the repaired solution.

$$f_\mathrm{fitness}(\boldsymbol{x}) = f(\boldsymbol{x}_\mathrm{repaired}) + \alpha \, \|\boldsymbol{x} - \boldsymbol{x}_\mathrm{repaired}\|^2 \ . \tag{51}$$

The repaired solution is disregarded afterwards.

In case of box-boundaries, $\boldsymbol{x}_{\text{repaired}}$ is set to the feasible solution with the smallest distance $\|\boldsymbol{x} - \boldsymbol{x}_{\text{repaired}}\|$. In other words, components that are infeasible in $\boldsymbol{x}$ are set to the (closest) boundary value in $\boldsymbol{x}_{\text{repaired}}$. A similar boundary handling with a component-wise adaptive α is described in [12].

No repair mechanism available The fitness of the infeasible search point $\boldsymbol{x}$ might similarly compute to

$$f_{\text{fitness}}(\boldsymbol{x}) = f_{\text{offset}} + \alpha \sum_i \mathbb{1}_{c_i>0} \times c_i(\boldsymbol{x})^2 \tag{52}$$

where, w.l.o.g., the (non-linear) constraints $c_i : \mathbb{R}^n \to \mathbb{R}, \boldsymbol{x} \mapsto c_i(\boldsymbol{x})$ are satisfied for $c_i(\boldsymbol{x}) \leq 0$, and the indicator function $\mathbb{1}_{c_i>0}$ equals to one for $c_i(\boldsymbol{x}) > 0$, zero otherwise, and $f_{\text{offset}} = \text{median}_k f(\boldsymbol{x}_k)$ equals, for example, to the median or 25%-tile function value of the feasible points in the same generation. If no other information is available, $c_i(\boldsymbol{x})$ might be computed as the squared distance of $\boldsymbol{x}$ to the best feasible solution in the population. This approach has not yet been experimentally evaluated by the author. A different, slightly more involved approach is given in [6].

In either case of (51) and (52), α should be chosen such that the differences in f and the differences in the second summand have a similar magnitude.

C MATLAB Source Code

```
function xmin=purecmaes
  % CMA-ES: Evolution Strategy with Covariance Matrix Adaptation for
  % nonlinear function minimization.
  %
  % This code is an excerpt from cmaes.m and implements the key parts
  % of the algorithm. It is intendend to be used for READING and
  % UNDERSTANDING the basic flow and all details of the CMA *algorithm*.
  % Computational efficiency is sometimes disregarded.

  % --------------------  Initialization --------------------------------

  % User defined input parameters (need to be edited)
  strfitnessfct = 'felli'; % name of objective/fitness function
  N = 10;               % number of objective variables/problem dimension
  xmean = rand(N,1);    % objective variables initial point
  sigma = 0.5;          % coordinate wise standard deviation (step-size)
  stopfitness = 1e-10;  % stop if fitness < stopfitness (minimization)
  stopeval = 1e3*N^2;   % stop after stopeval number of function evaluations

  % Strategy parameter setting: Selection
  lambda = 4+floor(3*log(N));  % population size, offspring number
  mu = lambda/2;   % lambda=12; mu=3; weights = ones(mu,1); would be (3_I,12)-ES
  weights = log(mu+1/2)-log(1:mu)';     % muXone recombination weights
  mu = floor(mu);      % number of parents/points for recombination
  weights = weights/sum(weights);       % normalize recombination weights array
  mueff=sum(weights)^2/sum(weights.^2); % variance-effective size of mu

  % Strategy parameter setting: Adaptation
  cc = (4+mueff/N) / (N+4 + 2*mueff/N);  % time constant for cumulation for C
  cs = (mueff+2)/(N+mueff+5);  % t-const for cumulation for sigma control
  c1 = 2 / ((N+1.3)^2+mueff);  % learning rate for rank-one update of C
  cmu = 2 * (mueff-2+1/mueff) / ((N+2)^2+2*mueff/2);  % and for rank-mu update
  damps = 1 + 2*max(0, sqrt((mueff-1)/(N+1))-1) + cs; % damping for sigma

  % Initialize dynamic (internal) strategy parameters and constants
  pc = zeros(N,1); ps = zeros(N,1);   % evolution paths for C and sigma
  B = eye(N);                         % B defines the coordinate system
  D = eye(N);                         % diagonal matrix D defines the scaling
  C = B*D*(B*D)';                     % covariance matrix
  eigeneval = 0;                      % B and D updated at counteval == 0
  chiN=N^0.5*(1-1/(4*N)+1/(21*N^2));  % expectation of
                                      %   ||N(0,I)|| == norm(randn(N,1))

  % -------------------- Generation Loop --------------------------------

  counteval = 0;  % the next 40 lines contain the 20 lines of interesting code
  while counteval < stopeval

    % Generate and evaluate lambda offspring
    for k=1:lambda,
      arz(:,k) = randn(N,1);  % standard normally distributed vector
      arx(:,k) = xmean + sigma * (B*D * arz(:,k));   % add mutation    % Eq. 37
      arfitness(k) = feval(strfitnessfct, arx(:,k)); % objective function call
      counteval = counteval+1;
    end

    % Sort by fitness and compute weighted mean into xmean
    [arfitness, arindex] = sort(arfitness); % minimization
    xmean = arx(:,arindex(1:mu))*weights;   % recombination            % Eq. 39
    zmean = arz(:,arindex(1:mu))*weights;   % == D^-1*B'*(xmean-xold)/sigma

    % Cumulation: Update evolution paths
    ps = (1-cs)*ps + (sqrt(cs*(2-cs)*mueff)) * (B * zmean);            % Eq. 40
    hsig = norm(ps)/sqrt(1-(1-cs)^(2*counteval/lambda))/chiN < 1.4+2/(N+1);
    pc = (1-cc)*pc + hsig * sqrt(cc*(2-cc)*mueff) * (B*D*zmean);       % Eq. 42

    % Adapt covariance matrix C
    C = (1-c1-cmu) * C ...                 % regard old matrix         % Eq. 43
         + c1 * (pc*pc' ...                % plus rank one update
                 + (1-hsig) * cc*(2-cc) * C) ...  % minor correction
         + cmu ...                         % plus rank mu update
           * (B*D*arz(:,arindex(1:mu))) ...
           *  diag(weights) * (B*D*arz(:,arindex(1:mu)))';

    % Adapt step-size sigma
    sigma = sigma * exp((cs/damps)*(norm(ps)/chiN - 1));               % Eq. 41

    % Update B and D from C
    if counteval - eigeneval > lambda/(cone+cmu)/N/10  % to achieve O(N^2)
      eigeneval = counteval;
      C=triu(C)+triu(C,1)'; % enforce symmetry
      [B,D] = eig(C);       % eigen decomposition, B==normalized eigenvectors
      D = diag(sqrt(diag(D))); % D contains standard deviations now
    end

    % Break, if fitness is good enough
    if arfitness(1) <= stopfitness
      break;
    end

    % Escape flat fitness, or better terminate?
    if arfitness(1) == arfitness(ceil(0.7*lambda))
      sigma = sigma * exp(0.2+cs/damps);
      disp('warning: flat fitness, consider reformulating the objective');
    end

    disp([num2str(counteval) ': ' num2str(arfitness(1))]);

  end % while, end generation loop

  % -------------------- Final Message ---------------------------------

  disp([num2str(counteval) ': ' num2str(arfitness(1))]);
  xmin = arx(:, arindex(1)); % Return best point of last generation.
                             % Notice that xmean is expected to be even
                             % better.

% ---------------------------------------------------------------
function f=felli(x)
  N = size(x,1); if N < 2 error('dimension must be greater one'); end
  f=1e6.^((0:N-1)/(N-1)) * x.^2;  % condition number 1e6
```

D Reformulation of Learning Parameter c_{cov}

For sake of consistency and clarity, we have reformulated the learning coefficients in (43) and replaced

$$\frac{c_{\mathrm{cov}}}{\mu_{\mathrm{cov}}} \quad \text{with} \quad c_1 \tag{53}$$

$$c_{\mathrm{cov}}\left(1-\frac{1}{\mu_{\mathrm{cov}}}\right) \quad \text{with} \quad c_\mu \quad \text{and} \tag{54}$$

$$1-c_{\mathrm{cov}} \quad \text{with} \quad 1-c_1-c_\mu \ , \tag{55}$$

and choosen (in replacing (49))

$$c_1 \;=\; \frac{2}{(n+1.3)^2+\mu_{\mathrm{cov}}} \tag{56}$$

$$c_\mu \;=\; \min\left(2\,\frac{\mu_{\mathrm{cov}}-2+\frac{1}{\mu_{\mathrm{cov}}}}{(n+2)^2+\mu_{\mathrm{cov}}}\,,\,1-c_1\right) \ , \tag{57}$$

The resulting coefficients are quite similar to the previous. In contrast to the previous formulation, c_1 becomes monotonic in μ_{eff}^{-1} and c_1+c_μ becomes virtually monotonic in μ_{eff}.

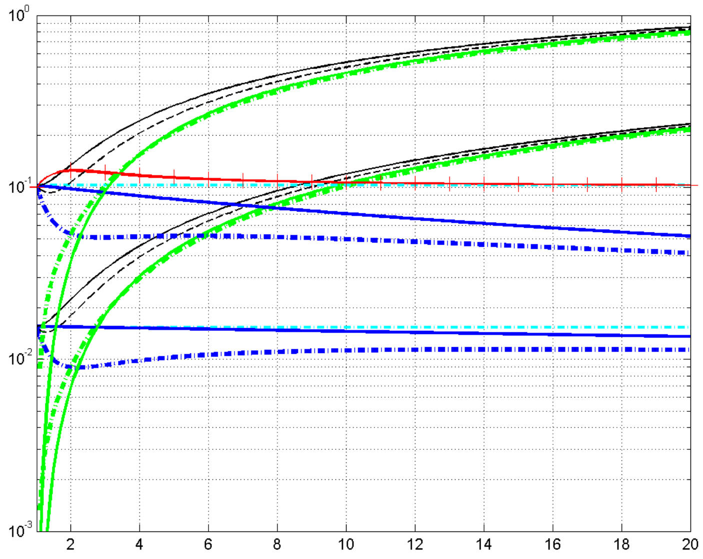
Another alternative, depending only on the degrees of freedom in the covariance matrix and additionally correcting for very small λ, reads

$$c_1 \;=\; \frac{\min(1,\lambda/6)}{m+2\sqrt{m}+\frac{\mu_{\mathrm{eff}}}{n}} \tag{58}$$

$$c_\mu \;=\; \min\left(1-c_1\,,\,\frac{\alpha_\mu^0+\mu_{\mathrm{eff}}-2+\frac{1}{\mu_{\mathrm{eff}}}}{m+4\sqrt{m}+\frac{\mu_{\mathrm{eff}}}{2}}\right) \tag{59}$$

$$\alpha_\mu^0 \;=\; 0.3 \ , \tag{60}$$

where $m=\frac{n^2+n}{2}$ is the degrees of freedom in the covariance matrix. For $\mu_{\mathrm{eff}}=1$, the coefficient c_μ is now chosen to be larger than zero, as $\alpha_\mu^0>0$. Figure 7 compares the new learning rates with the old ones.



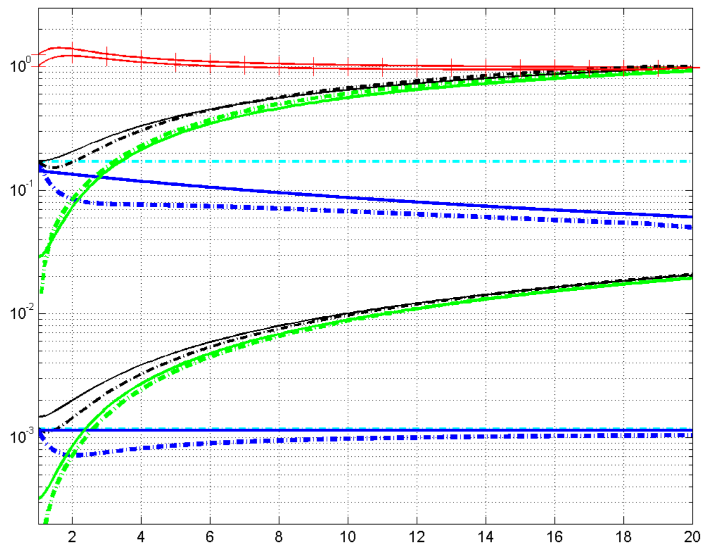


Figure 7: Learning rates c_1, c_μ (solid) and c_{cov} (dash-dotted) versus μ_{eff}. Above: Equations (56) etc. for $n = 3; 10$. Below: Equations (58) etc. for $n = 2; 40$. Black: $c_1 + c_\mu$ and c_{cov}; blue: c_1 and $c_{\mathrm{cov}}/\mu_{\mathrm{cov}}$; green: c_μ and $(1 - 1/\mu_{\mathrm{cov}})c_{\mathrm{cov}}$; cyan: $2/(n^2 + \sqrt{2})$; red: $(c_1 + c_\mu)/c_{\mathrm{cov}}$, above divided by ten. For $\mu_{\mathrm{cov}} \approx 2$ the difference is maximal, because c_1 decreases much slower with increasing μ_{cov} and c_{cov} is non-monotonic in μ_{cov} (a main reason for the new formulation).