LS-CMA-ES: a Second-order algorithm for Covariance Matrix Adaptation

Anne Auger, Marc Schoenauer, Nicolas Vanhaecke

LS-CMA-ES: a Second-order algorithm for Covariance Matrix Adaptation

Anne Auger[1], Marc Schoenauer[1], Nicolas Vanhaecke[2]

[1] TAO team, INRIA Futurs
LRI, Bât. 490, Université Paris-Sud
91405 Orsay Cedex, France
[2] Molecular Physics Department
Fritz-Haber-Institut der Max-Planck-Gesellschaft
Faradayweg 4-6, 14195 Berlin, Germany
Anne.Auger@lri.fr, Marc.Schoenauer@inria.fr, vanhaeck@fhi-berlin.mpg.de

Abstract. Evolution Strategies, Evolutionary Algorithms based on Gaussian mutation and deterministic selection, are today considered the best choice as far as parameter optimization is concerned. However, there are multiple ways to tune the covariance matrix of the Gaussian mutation. After reviewing the state of the art in covariance matrix adaptation, a new approach is proposed, in which the covariance matrix adaptation method is based on a quadratic approximation of the target function obtained by some Least-Square minimization. A dynamic criterion is designed to detect situations where the approximation is not accurate enough, and original Covariance Matrix Adaptation (CMA) should rather be directly used. The resulting algorithm is experimentally validated on benchmark functions, performing much better than CMA-ES on a large class of problems.

1 Introduction

Among the class of Evolutionary Algorithms (EAs), Evolution Strategies (ESs), based on the so-called Gaussian mutations, are today considered the state-of-the-art in parameter optimization, i.e. optimization of a function defined on $\mathbb{R}^n$ for some $n \geq 1$ [4]. Basic ESs generate λ offspring by mutating the μ parents (without selection), and then deterministically replace the parents by the μ best individuals from either the offspring (this variant is then called the $(\mu, \lambda) - ES$), or the merge of parents and offspring (and it is then called the $(\mu + \lambda) - ES$). The Gaussian mutation operator for this large class of algorithm consists in adding some Gaussian random noise to the parent, and can be formally written as

$$\mu_{\sigma,C} : X_0 \mapsto X_0 + \sigma N(0, C). \tag{1}$$

where $N(0, C)$ represents one realization of a multivariate Gaussian distribution with mean 0 and *covariance matrix* C, a symmetric positive definite matrix, and σ is a positive real value called the *step-size*. This distinction can seem arbitrary

(the actual covariance matrix is σC), but it is often convenient to consider that C is somehow normalized, giving the "direction" in which sampling should occur, while σ indicates how far the mutation should go in that "direction".

The performance of the EA is of course highly dependent on the choices of C and σ, that have to be tuned not only to the problem at hand, but also to the current state of the evolution process. Several techniques have been proposed, from the self-adaptation of the mutation parameters in Evolution Strategies (SA-ES) [11] to the Covariance Matrix Adaptation (CMA-ES) [7] where the covariance matrix C is deterministically adapted from the last move of the algorithm. Some Estimation of Distribution Algorithms [8] can also be considered as pertaining to the same category of algorithms that repeatedly sample some Gaussian distribution to generate next test points.

However, we argue in this paper (Section 2) that none of those algorithms does make an optimal use of all previously sampled points . We then discuss, in section 3, what the optimal covariance matrix should be, based on the case of elliptic functions. In Section 4 we propose the LS-CMA-ES algorithm, an ES-like algorithm that updates the covariance matrix by learning some curvature information of the fitness function. Finally, Section 5 presents and discusses experimental results on several benchmark functions from the literature, while section 6 derives from those results further work that must be done to extend the proposed method (e.g. to better handle multi-modal cases) and to complete its assessment.

2 State of the Art in Covariance Matrix Learning

Since the early work of Rechenberg and Schwefel, who first used Gaussian mutations to evolve real-valued parameters (see e.g. [3] for a detailed history of the field), it has been clear that the most critical issue of ESs was the tuning of the mutation parameters. Initial works were concerned mainly with the step-size: Rechenberg's one-fifth rule [9], derived from theoretical considerations on the progress rate, as well as Schwefel's first self-adaptive ES [11] were only concerned with the optimal step-size, i.e. considered the covariance matrix to be the identity matrix. However, Schwefel soon proposed to self-adapt one step-size per variable, i.e. to use a covariance matrix that is diagonal with positive diagonal terms, and further extended the self-adapted parameters to the so-called *correlated mutations*, corresponding to the general case of Equation 1 where all parameters of the covariance matrix C are self-adapted.

The ground idea of self-adaptation for correlated mutations is to rely on mutations of the mutation parameter themselves to adjust the covariance matrix C. Whereas this clever idea removed the need to manually tune the mutation parameters, it does not take into account the history of evolution to direct the search – as did the one-fifth rule for the step-size. From thereon, Hansen and Ostermeier designed the Covariance Matrix Adaptation-ES (CMA-ES) algorithm

[7], later made more computationally efficient in [6]. The basic idea of CMA is to deterministically use the the most recent descent direction, i.e. the direction between the best parents at two consecutive generations: the covariance matrix C is gradually updated with rank-one matrices whose unique non-zero eigenvalue has as eigenvector the last descent direction (see the complete equations at lines 8 and 10 of Table 1). In some sense, whereas CMA has been thought as a derandomized correlated mutation from the ES point of view, it could also be viewed as a randomized steepest-descent algorithm from the point of view of standard numerical optimization. Note that the complete CMA algorithm also includes some derandomized step-size adaptation. However, it has been clearly experimentally demonstrated (not shown here for space reasons) that this step-size adaptation is not crucial, as the performances of the algorithms are roughly the same if the step-size is self-adapted using the standard SA-ES mechanism (see [11], or line 4 of Table 1, for the formal equation).

Estimation of Distribution Algorithms (EDAs) is a recent paradigm using similar ideas – at least in the continuous case (see [8] for a survey of the whole field). EDAs evolve a probability distribution over the search space. The main loop of an EDA first samples that distribution, then selects some of the samples according to their fitness, and finally updates the distribution from those selected individuals. Hopefully, the distribution will gradually concentrate on the optima of the fitness function. Of course, the update step depends on the model that has been chosen for the distribution. A typical EDA for parameter optimization is the $EMNA$ family [8], where the distribution is sought as a multi-variate Gaussian distribution: the sampling phase of $EMNA$ using such a distribution is then equivalent to the birth of offspring using Equation 1, and EMNA does adapt a covariance matrix, similarly to the ES algorithms described above.

On the one hand, EMNA uses many points to refine the covariance matrix – and hence one could think that it will get a better approximation of the "optimal" covariance matrix, if such a thing exists (see Section 3). However, the selection phase of EMNA (and of most EDAs) is merely binary: if for instance the best half of the population is selected, individuals get selected without any reference to their respective fitness. Moreover, in the case where a completely new covariance matrix is generated from the selected samples, using some Maximum Likelihood principle, such a covariance matrix only tries to "explain" the distribution of sample at hand, without paying any attention to other parts of the search space, even those that have been visited before. And this situation is only slightly improved if the covariance matrix is updated from the previous one at each generation.

Experiments have shown that the best-performing of those algorithms is clearly the CMA-ES algorithm (see [7] for comparison of SA-ES and CMA-ES, and [10] for a comparison of CMA-ES with some EDAs) – and the reason for that seems to be that the covariance matrix is better adapts to teh fitness landscape. The purpose of this paper is to try to make a better use of all the

points that have been sampled by the algorithm, when updating the covariance matrix, by learning the local curvature of the target function. But before that, we need to clarify what could be the ideal covariance matrix within an algorithm using Gaussian sampling.

3 Rationale

The simplest problem, on which the behavior of optimization algorithm can be studied in great detail, is the sphere model, where the goal is to minimize the function f_S defined by

$$f_s(x) = x^T x. \tag{2}$$

It has been shown numerically [4] that the optimal covariance matrix in that case is the identity matrix I_d. Moreover, some theoretical studies have proved the convergence of the on $(1, \lambda) - ES$, either with a dynamic step-size [2] or in the case of self-adaptive step-size [1]. From those results, plus the naive consideration of the isotropy of the fitness function itself, we shall from now on suppose that indeed I_d is a good, if not optimal, covariance matrix when optimizing the sphere function.

But suppose that we now want to minimize the elliptic function f_E

$$f_E(x) = \frac{1}{2} x^T H x, \tag{3}$$

where H is a symmetric positive definite matrix. The choice of the optimal covariance matrix to solve that problem with Evolution Strategies becomes obvious after a change of variables that will turn this problem into a sphere problem. Consider the eigen decomposition of H in an orthonormal basis:

$$H = P^T \Delta \Delta P \tag{4}$$

where P is an orthonormal matrix ($P^{-1} = P^T$) and Δ a diagonal matrix whose diagonal terms are the square roots of the eigenvalues of H. Now let $W = \Delta^{-1} P X$. Then simple calculations show that $f_E(X) = f_S(W)$ and that the mutation operator given by equation (1) with $C = (\frac{1}{2} H)^{-1}$ transforms W_0 into $W_0 + \sigma N(0, I_d)$. Hence, if we consider that I_d is the best choice for the covariance matrix for the sphere problem, then $(\frac{1}{2} H)^{-1}$ is the best choice for the covariance matrix for the elliptic problem.

For arbitrary functions, But on the other hand, thanks to Taylor expansion, all regular functions can be locally approximated by an elliptic function:

$$f(X) = f(X_0) + (X - X_0)^T \nabla f(X_0) + \frac{1}{2} (X - X_0)^T H(X_0)(X - X_0) + o(||X - X_0||^2) \tag{5}$$

where $\nabla f(X_0)$ denotes the gradient of f at point X_0 and $H(X_0)$ the Hessian matrix of f at point X_0. It hence makes sense to try to use within ESs an

approximation of the Hessian matrix of the function at hand within the mutation operator. But before proposing a deterministic algorithm using points previously sampled during evolution to approximate that Hessian matrix, let us first look at existing methods to derive a covariance matrix in the evolutionary community.

4 The LS-CMA-ES algorithm

The goal of this section is to introduce an algorithm that will try to use an approximation of the Hessian matrix of the target function as advocated in section 3: This approximation will be deterministically built using the available sample points – and later used within the Gaussian mutation 1.

Whereas such strategy is optimal for elliptic functions, it can fail on functions that are "more flat" (e.g. the famous Rosenbrook function), because the quadratic approximation then becomes very inaccurate. A criterion needs to be designed to detect such cases and to alternatively use the CMA "steepest descent update rule" for the covariance matrix.

4.1 Approximating the Hessian matrix

This step of the proposed algorithm aims at learning the local Hessian matrix of the target function. Starting from the Taylor expansion (see equation 5), and supposing that we have some sample points of values taken by f, i.e. a set of points X_k together with their actual fitness $f(X_k)$, located "not too far" from a given point X_0, approximations g and H for both $\nabla f(X_0)$ and $H(X_0)$ can be easily obtained by solving the linear least-square minimization problem:

$$\min_{g\in\mathbb{R}^d, H\in\mathcal{S}(\mathbb{R}^d)} \sum_{k=1}^{N} \Big(f(X_k) - f(X_0) - (X_k - X_0)^T g - \frac{1}{2}(X_k - X_0)^T H (X_k - X_0) \Big)^2 \quad (6)$$

The unknowns of this problem are g (n unknown parameters), and H ($n(n+1)/2$ unknown parameters). As soon as more than $n(n+3)/2$ sample points are available, the linear system corresponding to equation 6 is overdetermined: the solution is found by evaluating the pseudo-inverse of this linear system. The cost of the direct computation of this pseudo-inverse by standard numerical methods (e.g. directly available in Matlab) is scaling as n^6.

Note that if the function f is elliptic, the solution of problem 6 is exactly given by $\nabla f(X_0)$ and $H(X_0)$, and the least-square value reaches 0. However, in the general case, denoting by $\hat{g}$ and $\hat{H}$ the solutions of problem 6, the minimum is not 0. Moreover, a measure of the quality of the approximation of the gradient and the Hessian matrix can be given by

$$\sum_{k=1}^{N} \Big(f(X_k) - f(X_0) - (X_k - X_0)^T \hat{g} - \frac{1}{2}(X_k - X_0)^T \hat{H} (X_k - X_0) \Big)^2 .$$

More precisely, a criterion to detect the cases of poor approximations where matrix $\hat{H}$ should not be used in the mutation will be based on the following normalized error:

$$\mathcal{Q} = \frac{1}{N}\sum_{k=1}^{N}\Big(\frac{f(X_k) - f(X_0) - (X_k - X_0)^T\hat{g} - \frac{1}{2}(X_k - X_0)^T\hat{H}(X_k - X_0)}{f(X_k) - f(X_0) - (X_k - X_0)^T\hat{g}}\Big)^2. \tag{7}$$

This Q factor is invariant under any dilatation or offset of the target function f.

4.2 Gathering examples

The approximation method described above requires at least $n(n^2+3)/2$ samples (i.e. points of the search space together with their actual fitnesses). Those samples will of course be gathered from the points used during evolution. A trade-off has to be found, however, between the accuracy of the approximation and the computational effort, as drawing exactly $n(n^2+3)/2$ points at every generation would obviously be far too costly. Hence only points that need to be evaluated during the normal course of the algorithm will be used, and the approximation will be based on the most recent visited points. The approximation could nevertheless be computed at every generation, using some sliding window of recent points. But again, this has a cost, and another trade-off has to be made, computing a new approximation only at given times, using the same covariance matrix in between (this is similar to what is done in most pseudo-Newton methods).

4.3 Adapting the step-size

So far, we have only discussed the covariance matrix of the Gaussian mutation described by Equation 1. However, even with the optimal covariance matrix, an ES algorithm will never work properly if the step-size is not correctly adapted too. As mentioned in section 2, CMA-ES has a deterministic update rule for its step-size, but performs as well with self-adaptive step-size. Moreover, the transition between Least-Square phases and CMA-phases of the algorithm makes it difficult to mix the different strategies for the step-size. It was hence decided to use the standard self-adaptive rule of ES algorithms for the step-size throughout the LS-CMA-ES algorithm.

4.4 The $(1, \lambda)$-LS-CMA-ES

The first complete algorithm based on the ideas developed above is the $(1, \lambda)$-LS-CMA-ES algorithm, whose pseudo-code is given in Table 1, where g is the generation counter, $x^{(g)}$ the current parent, $\sigma^{(g)}$ the current step-size, $C^{(g)}$ the current covariance matrix, and c_{cov} a real parameter that will tune the covariance matrix update rule. We shall now discuss the different steps in turn.

A $(1, \lambda)$**-SA-ES**. The basis of the LS-CMA algorithm is a standard self-adaptive Evolution Strategy: Line 4 of Table 1 is the generation of λ offspring using the

Gaussian mutation of equation 1 where- the step-size is a log-normal mutation of the step-size of the parent. Lines 5 and 6 are the usual evaluation and selection steps of $(1, \lambda)$-ES.

Covariance matrix update. Lines 8 and 10 are the standard covariance matrix adaptation of the CMA method [6]. However, the computation of the steepest direction p_c must be performed even in LS mode to be accurate if mode must be switched. In LS mode, the covariance matrix of the mutation is unchanged (line 9).

Quadratic approximation. Every n_{upd} generations (line 11), the approximation of the Hessian is computed from equation 6 (line 12). The sample points have been gathered along evolution (line 7).

If the error on the approximation is below the threshold, then the new approximation replaces the old one, whether it came from CMA or LS modes, and mode is turned to LS (line 13). Otherwise, mode is turned to CMA (line 14). And the main loop resumes (line 16).

Initialization: $x^{(0)} \leftarrow x_0$; $\sigma^{(0)} \leftarrow \sigma_0$; $C^{(0)} \leftarrow I_d$; $p_c^{(0)} \leftarrow 0$; archive $\leftarrow \emptyset$ Mode $\leftarrow$ LS	1
while (**not** stopping condition) **do**	2
$g \leftarrow g + 1$	3
Create λ offspring: $x_j^{(g+1)} \leftarrow x^{(g)} + \sigma^{(g)} \exp(\tau \tilde{N}_j(0,1)) N_j(0, C^{(g)})$, $j \in [1, \lambda]$	4
Evaluate offspring: Compute $f(x_j^{(g+1)})$ for all $1 \leq j \leq \lambda$	5
Select best offspring $x^{(b)}$: $x^{(g+1)} \leftarrow x_b^{(g+1)}$; $\sigma^{(g+1)} \leftarrow \sigma^{(g)} \exp(\tau \tilde{N}_b(0,1))$	6
Store offspring in archive	7
$p_c^{(g+1)} \leftarrow (1 - c_c) p_c^{(g)} + \frac{\sqrt{c_c(2-c_c)}}{\sigma^{(g)}} (x^{(g+1)} - x^{(g)})$	8
if in LS mode $C^{(g+1)} \leftarrow C^{(g)}$	9
else (CMA mode) $C^{(g+1)} \leftarrow (1 - c_{cov}) C^{(g)} + c_{cov} p_c^{(g+1)} (p_c^{(g+1)})^T$	10
if (g mod n_{upd} **=0)**	11
Compute $\hat{g}_{x^{(g)}}$, $\hat{H}_{x^{(g)}}$ from n^2 recent archived samples solving Eq. 6	12
if $(\mathcal{Q}(\hat{g}_{x^{(g)}}, \hat{H}_{x^{(g)}}) < \mathcal{Q}_{th}$ **mode** $\leftarrow$ **LS**; $C^{(g+1)} \leftarrow (\frac{1}{2} \hat{H}_{x^{(g)}})^{-1}$	13
else **mode** $\leftarrow$ **CMA**	14
end if	15
end while	16

Table 1. Pseudo code for the $(1, \lambda)$-LS-CMA-ES algorithm

The parameters. The values of some parameters of the LS-CMA-ES algorithm have to be manually fixed. The following values, either from [11] or [6] when relevant, or adjusted by trial-and-error during preliminary runs, have been used in all experiments of Section 5. The number of offspring λ was set to 10 for the LS-CMA-ES and to $4 + \lfloor 3 * \log(N) \rfloor$ for the CMA algorithm, c_c and c_{cov}, the relaxation parameters, to respectively $\frac{4}{n+4}$ and $\frac{2}{(n+\sqrt{2})^2}$ [6]. The update parameter of the self-adaptive log-normal mutation of the step-size, τ, was set to $\frac{1}{\sqrt{n}}$ [11]. The threshold $\mathcal{Q}_{th}$ on the approximation error for the Hessian matrix

that will decide to switch back and forth between LS mode and pure CMA mode, was set to 10^{-3}, after intensive monitoring on the elliptic and Rosenbrook functions. Finally, n_{upd} was set to 100 with of course a possibly delay in the first update in order to have n^2 samples available for the first update.

5 Experimental Results

This section presents the first results obtained using the $(1, \lambda)$-LS-CMA-ES algorithm. It will be compared to the best-performing to-date algorithm in the class of ES-like algorithms (i.e. relying on repeated Gaussian mutations), the original CMA algorithm, as described in [6], using the original implementation that the first author kindly makes available [5]. Two variants of CMA are used, the best performing CMA, that uses $\lfloor \frac{\lambda}{2} \rfloor$ parents as default value, and a $(1, \lambda)$-CMA, for a fair comparison, as no experiment has yet been performed on multi-membered LS-CMA-ES. The analytical form of all test function is given in Table 5.

Results: First, the elliptic function f_{elli} and the Rosenbrock function f_{Ros} have been intensively tested. A priori, if our hypothesis of section 3 about the optimality of $(\frac{1}{2}H)^{-1}$ as covariance matrix is correct, the LS-CMA-ES method should perform very well on the elliptic function. Rosenbrock function, on the other hand, is quite different, and behaves more like $|x|^6$ close to the optimum. The approximation error $\mathcal{Q}$ defined in Equation 7 should hence be larger.

All tests have been performed 100 times independently, from the same initial position $(5, 5, \ldots, 5)$. At each generation, the minimum, median and maximum values out of the 100 runs have been gathered, and are shown on Figure 1.

The results for f_{elli} are very clear, and match our expectations: the three groups of three different lines, from left to right are respectively the min, median and max for LS-CMA-ES, then the min, median and max for pure CMA and finally the min, median and max for $(1, \lambda)$-CMA. The variance is very small for all three algorithms. For both CMA-ES algorithms, the flat plateau before actually diving toward 0 is the time necessary to learn the correct covariance matrix – while LS-CMA-ES has it all right from the beginning, whatever the dimension (only the minimum number of sample points before the first matrix can be learned makes a small difference). Note, however, that such plateaus can result in a huge difference of actual results in practice, when the stopping criterion is likely to be driven by the computational cost when the actual value of the minimum is generally unknown.

The picture is somewhat different with the Rosenbrock function – which is not unimodal, and has a very flat region between the local optimum and the global optimum. First, only the min and median values have been plotted. For both 20 and 50 dimensions, the first three lines from left to right are the min values of, respectively, LS-CMA-ES, pure CMA and $(1, \lambda)$-CMA. Next three lines are the median value for, respectively, the pure CMA, the $(1, \lambda)$-CMA and the LS-CMA-ES. As can be seen in dimension 50, the median run of LS-CMA-ES did not converge to the right minimum and the fitness value stayed above 1.

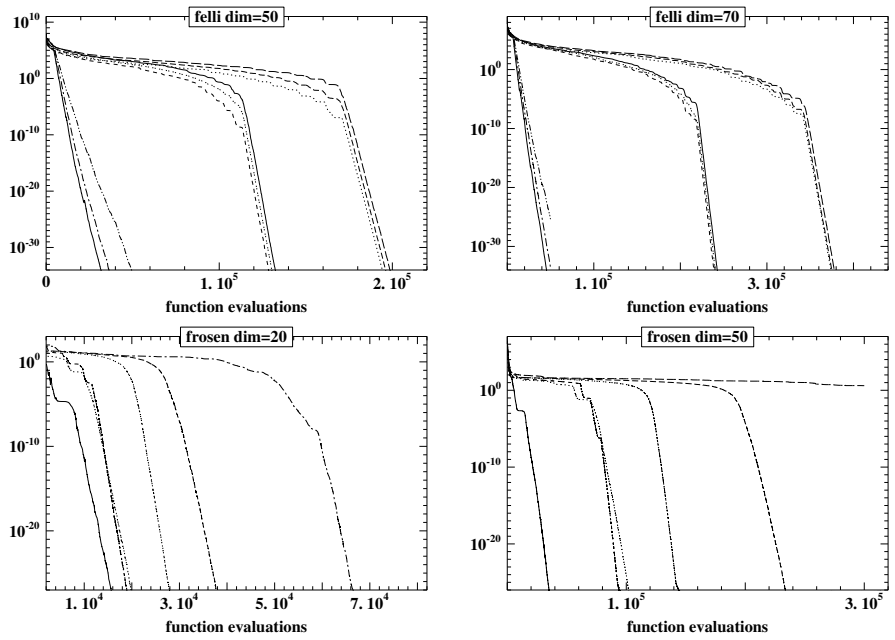

Fig. 1. Comparative on-line results for the elliptic function in dimensions 50 and 70 (above) and for the Rosenbrook function in dimensions 20 and 50 (below). See text for line labels.

Indeed, for all three algorithms, some of the 100 runs did not converge to the global optimum. More precisely, this concerns, for LS-CMA-ES, pure CMA and $(1, \lambda)$-CMA, 25 (resp. 12 and 22) runs in dimension 20, and 65 (resp. 15 and 20) runs in dimension 50.

Furthermore, a detailed investigation of the approximation error shows that LS is active at the beginning of evolution, far from the flat optimum, and boosts the search there. As the individuals get closer to the minimum, the approximation error increases, and at some point the algorithm switches to CMA, sometimes keeping its advance (e.g. for the runs who gave the min values), sometimes not. Some better tuning of the error threshold might prevent this to happen, but more detailed investigations are needed to understand why CMA catches up.

Several other functions, also used in [7, 6] (except for f_{exp}), have been investigated to further validate those first conclusions. Some off-line results are summarized in table 5. The tendencies observed on the elliptic and Rosenbrock functions are indeed confirmed: f_{cigar}, f_{tablet} and f_{cigtab} are elliptic functions, on which LS-CMA-ES definitely outperforms even the complete CMA; On the other hand, $f_{diff-pow}$ is and f_{exp} are not elliptic, f_{exp} is even "infinitely flat" (all derivatives are 0 at the minimum), and the results are similar to those on the Rosenbrock function in terms of switching between LS mode and CMA mode. However, as these functions are unimodal, no premature convergence takes place.

	LS-CMA-ES			CMA-ES			$(1, \lambda) - CMA$		
Function	min	med	max	min	med	max	min	med	max
$f_{elli} = \sum_{i=1}^{n} (10^6)^{\frac{i-1}{n-1}} x_i^2$	0.51	0.57	0.72	2.05	2.15	2.26	2.9	3.0	3.1
$f_{Ros} = \sum_{i=1}^{n-1} 100(x_i^2 - x_{i+1})^2 + (x_i - 1)^2$	0.98	6.03	–	1.4	2.3	–	1.4	3.1	–
$f_{cigtab} = x_1^2 + \sum_{2}^{n-1} 10^4 x_i^2 + 10^8 x_n^2$	0.53	0.61	0.70	1.45	1.52	1.58	2.36	2.47	2.57
$f_{tablet} = 10^6 x_1^2 + \sum_{2}^{n} x_i^2$	0.42	0.50	0.56	1.61	1.71	1.78	3.4	3.7	3.9
$f_{cigar} = x_1^2 + \sum_{2}^{n} 10^6 x_i^2$	0.45	0.51	0.57	0.90	0.95	1.01	1.04	1.08	1.1
$f_{diff-pow} = \sum_{1}^{n} \lvert x_i \rvert^{2+10\frac{i-1}{n-1}}$	1.53	2.14	3.66	1.18	1.36	1.58	1.53	2.14	2.66
$f_{exp} = \exp(\lVert x \rVert^2) - 1$	0.22	0.29	0.31	0.27	0.29	0.32	0.35	0.40	0.45

Table 2. Off-line results for different test functions in dimension 20. The figures are the number of function evaluations (unit = 10^4) before the algorithm reached 10^{-10} (or – when it never did within 10^5 evaluations.

Discussion: Those experimental results show an outstanding improvement of performance for LS-CMA-ES over pure CMA-ES on elliptic functions. This should not be too surprising if referring to classical methods of numerical optimization: Whereas the CMA-ES can be seen as a steepest descent technique, where the covariance matrix is updated using the previous descent direction (see Section 2), the LS-CMA-ES can be considered similar to second-order quasi-Newton techniques, i.e. relying on local curvature information to find out next points to sample.

Yet, the improvement has only be demonstrated on unimodal or quasi-unimodal functions – and the niche for Evolutionary Algorithms is global optimization in multi-modal context. However, the exploitation phase of EAs is important too, and many refinements of ES algorithms have concentrated on that. As for exploration, it is generally emphasized in the early generations, when the step-sizes are still large enough, by using more than one parent. At the moment, only $(1, \lambda)$-LS-CMA-ES have been experimented with, and for few values of λ. It is expected that using multiple parents within LS-CMA-ES will increase its performance, for multi-modal problems, of course, including the Rosenbrock function, but also on non-elliptic unimodal problems, where at the moment if performs better than the corresponding $(1, \lambda)$-CMA.

One particular aspect of the LS-CMA-ES that cannot be seen on the plots of Section 5 is the computational cost: as mentioned in Section 4.1, direct solution of the least-square linear system theoretically scales like n^6. Thanks to Matlab efficient implementation, it was nevertheless possible to go up to dimension 70 (overhead costs for one approximation on a 3.0GHz Pentium roughly take 1, 280 and 1320 seconds for dimensions 20, 50 and 70). A first remark is that in high dimensions, this additional cost could be greatly reduced by using an iterative method to solve Equation 6. Furthermore, when solving very costly real-world problems, e.g. in CFD domain, where the computation of a single fitness value can take up to a few hours, the approximation overhead will be negligible compared to the gain of a few fitness computation.

6 Conclusion

This paper has introduced LS-CMA-ES, the first second-order evolutionary algorithm for parameter optimization. Initial results have demonstrated the efficiency of this algorithm on a large class of (quasi-)unimodal functions, where the best CMA-ES algorithm is outperformed by several orders of magnitude on elliptic problems. Moreover, a dynamic criterion allows the algorithm to switch back to CMA when the second-order approximation is poor, resulting in better results than the one-parent CMA-ES on "more flat" functions, as far as the best runs are concerned. There remains a lot of room for improvement of the proposed algorithm. First, its parameters (e.g. the number of offspring) needs to be fine-tuned. Next, there might exist some CMA-like update rule for the step-size to tackle premature convergence problems like observed on Rosenbrock function. Finally, the multi-membered version of the algorithm will be investigated, to address multi-modal problems. We do believe that this algorithm is a significant step forward in Evolutionary Parameter Optimization, and will be able to combine the accuracy of the best deterministic methods with the global search capabilities of Evolutionary Algorithms.

References

1. A. Auger. Convergence Results for (1,λ)-SA-ES using the Theory of φ-irreducible Markov Chains. In Th. Bäck and M. Schoenauer, editors, *Wokshop on Evolutionary Algorithms – ICALP 2003*, Eindhoven, The Netherlands, July 2003.
2. A. Auger, C. Le Bris, and M. Schoenauer. Dimension-independent Convergence Rate for Non-isotropic $(1, \lambda)-$ES. In E. Cantu-Paz et al., editor, *GECCO'2003*, pages 512–524. LNCS 2723 and 2724, Springer Verlag, 2003.
3. T. Bäck, F. Hoffmeister, and H.-P. Schwefel. A survey of Evolution Strategies. In R. K. Belew, L. B. Booker, editors, *ICGA'91*, pages 2–9. Morgan Kaufmann, 1991.
4. Th. Bäck and H.-P. Schwefel. An Overview of Evolutionary Algorithms for Parameter Optimization. *Evolutionary Computation*, 1(1):1–23, 1993.
5. N. Hansen. CMA-ES for Noisy Optimization: Implementations in Matlab. `http://www.bionik.tu-berlin.de/user/niko/cmaes_inmatlab.html`.
6. N. Hansen, S. Müller, and P. Koumoutsakos. Reducing the Time Complexity of the Derandomized Evolution Strategy with Covariance Matrix Adaptation (CMA-ES). *Evolutionary Computation*, 11(1):1–18, 2003.
7. N. Hansen and A. Ostermeier. Completely Derandomized Self-Adaptation in Evolution Strategies. *Evolutionary Computation*, 9(2):159–195, 2001.
8. P. Larranaga and J. A. Lozano. *Estimation of Distribution Algorithms. A New Tool for Evolutionary Computation*. Kluwer Academic Publishers, 2001.
9. I. Rechenberg. *Evolutionstrategie: Optimierung Technisher Systeme nach Prinzipien des Biologischen Evolution*. Fromman-Hozlboog Verlag, Stuttgart, 1973.
10. N. Hansen D. Büche J. Ocenasek S. Kern, S.D. Müller and P. Koumoutsakos. Learning Probability Distributions in Continuous Evolutionary Algorithms - A Comparative Review. In Th. Bäck and M. Schoenauer, editors, *Wokshop on Evolutionary Algorithms – ICALP 2003*, Eindhoven, The Netherlands, July 2003.
11. H.-P. Schwefel. *Numerical Optimization of Computer Models*. John Wiley & Sons, New-York, 1981. 1995 – 2^{nd} edition.