Benchmarking a Weighted Negative Covariance Matrix Update on the BBOB-2010 Noiseless Testbed

Nikolaus Hansen, Raymond Ros

To cite this version:

HAL Id: hal-00545728
https://hal.archives-ouvertes.fr/hal-00545728v2
Submitted on 13 Dec 2010

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
ABSTRACT

We implement a weighted negative update of the covariance matrix in the CMA-ES—weighted active CMA-ES or, in short, aCMA-ES. We benchmark the IPOP-aCMA-ES and compare the performance with the IPOP-CMA-ES on the BBOB-2010 noiseless testbed in dimensions between 2 and 40. On nine out of 12 essentially unimodal functions, the aCMA is faster than CMA, in particular in larger dimension. On at least three functions it also leads to a (slightly) better scaling with the dimension. In none of the 24 benchmark functions aCMA appears to be significantly worse in any dimension. On two and five functions, IPOP-CMA-ES and IPOP-aCMA-ES respectively exceed the record observed during BBOB-2009.

Categories and Subject Descriptors
G.1.6 [Numerical Analysis]: Optimization—global optimization, unconstrained optimization; F.2.1 [Analysis of Algorithms and Problem Complexity]: Numerical Algorithms and Problems

General Terms
Algorithms, performance, comparison

Keywords
CMA-ES, IPPOP-CMA-ES, active CMA-ES, Benchmarking, Black-box optimization

1. INTRODUCTION

The Covariance Matrix Adaptation Evolution Strategy (CMA-ES) [12, 11, 10] is a stochastic search procedure that samples new candidate solutions from a multivariate normal distribution thereof mean and covariance matrix are adapted after each iteration. The (µ/µw, λ)-CMA-ES samples new candidate solutions and selects the µ best among them. They contribute in a weighted manner to the update of the distribution parameters. The algorithm is non-elitist by nature, but a practical implementation will preserve the best-ever evaluated solution. Elitist variants of the CMA-ES [13] are often slightly faster but more susceptible of getting stuck in a suboptimal local optimum.

The IPOP-CMA-ES [1] implements a restart procedure. Before each restart, λ is doubled. In most cases, doubling λ increases the length of single runs, but it often improves the quality of the best found solution. The BIPOP-CMA-ES [6], proposed recently, maintains two budgets. Under the first budget, an IPPOP-CMA-ES is executed. Under the second budget, a multi-start (µ/µw, λ)-CMA-ES with various small population sizes is entertained.

Previous Benchmarking of CMA-ES. The (µ/µw, λ)-CMA-ES has been quite successful in several elaborate benchmarking exercises. Notably, the IPOP-CMA-ES [1] in the special session on real parameter optimization CEC-05 and the BIPOP-CMA-ES [6] in the black-box optimization benchmarking BBOB-2009 have shown excellent performance [5, 8] on respectively 25 and 24 uni- and multi-modal benchmark functions in search space dimension up to 50.

A comparison of two Estimation of Distribution Algorithms and three Evolution Strategies is presented in [15] on 14 functions. The CMA-ES performed best on 11 functions with a median speed-up by a factor of at least 30 compared to any other algorithm. On the remaining functions the maximum loss factor was 1.4.

A comparison on a small collection of multi-modal functions with three other algorithms is presented in [10]. The CMA-ES with optimal population size performs clearly best on the non-separable functions and is clearly outperformed on additively decomposable functions by Differential Evolution (DE).

A comparison of derivative-free optimization methods and BFGS on smooth unimodal functions is presented in [2]. On these functions, PSO and DE are in general significantly slower than CMA-ES. Only PSO performs similar on separable problems. NEWUOA and BFGS remarkably outperform CMA-ES, if the problem is convex and has a moderate conditioning. On non-convex problems with at least moderate condition number (i.e. 10^3) and on non-separable problems with higher condition number (i.e. 10^6), the CMA-ES breaks even and becomes advantageous with increasing condition number, tested up to 10^10.

In [3] the CMA-ES is recognized as state-of-the-art in

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

GECCO’10, July 7–11, 2010, Portland, Oregon, USA.
Copyright 2010 ACM 978-1-4503-0073-5/10/07 ...$10.00.

1http://www.ntu.edu.sg/home/EPNSugan/index_files/CEC-05/CEC05.htm
2http://coco.gforge.inria.fr/doku.php?id=bbob-2009
3http://www.scholarpedia.org/article/Evolution_
Given \( t \in \mathbb{N} \cup \{0\} \), \( \mathbf{m}^t \in \mathbb{R}^D \), \( \sigma^t \in \mathbb{R}_+ \), \( \mathbf{C}^t \in \mathbb{R}^{D \times D} \) positive definite, \( \mathbf{p}_m^t, \mathbf{p}_c^t \in \mathbb{R}^D \) and \( \mathbf{p}_0^t = \mathbf{p}_0^{t+0} = \mathbf{0} \), \( \mathbf{C}^{t+0} = \mathbf{I} \)

\[
\mathbf{x}_i \sim \mathbf{m}^t + \sigma^t \cdot \mathcal{N}(0, \mathbf{C}^t) \quad \text{is normally distributed for } i = 1, \ldots, \lambda
\]

\[
\mathbf{m}^{t+1} = \mathbf{m}^t + \sum_{i=1}^{\lambda} w_i (\mathbf{x}_{i,\lambda} - \mathbf{m}^t) \quad \text{where } f(\mathbf{x}_{1,\lambda}) \leq \cdots \leq f(\mathbf{x}_{\mu,\lambda}) \leq f(\mathbf{x}_{\mu+1,\lambda}) \ldots
\]

\[
\mathbf{p}_{a}^{t+1} = (1 - c_a) \mathbf{p}_a^t + \sqrt{c_a (2 - c_a) \mu_c} \mathbf{C}^t \cdot \frac{\mathbf{m}^{t+1} - \mathbf{m}^t}{c_m \sigma^t}
\]

\[
\lambda_a = \begin{cases} 1 & \text{if } \|\mathbf{p}_{a}^{t+1}\| < \sqrt{1 - (1 - c_a)^{2(t+1)}} (1 + \frac{2}{\sqrt{t+1}}) \|\mathbf{N}(0, \mathbf{I})\| \quad \text{stalls the update of } \mathbf{p}_a \text{ in (5) when } \sigma \\
\text{otherwise} & \end{cases}
\]

\[
\mathbf{p}_{c}^{t+1} = (1 - c_c) \mathbf{p}_c^t + \lambda_a \sqrt{c_c (2 - c_c) \mu_w} \frac{\mathbf{m}^{t+1} - \mathbf{m}^t}{c_m \sigma^t}
\]

\[
C_{\mu} = \sum_{i=1}^{\lambda} w_i \frac{(\mathbf{x}_{i,\lambda} - \mathbf{m}^t)^T}{\sigma^t} \quad \text{matrix with rank } \min(\mu, D) \text{ for the so-called rank-} \mu \text{ update of } \mathbf{C}^t
\]

\[
C_{\mu} = \sum_{i=0}^{\lambda} w_{i+1} \mathbf{y}_{i+1, \lambda} \mathbf{y}_{i+1, \lambda}^T \quad \text{with } \mathbf{y}_{i, \lambda} = \frac{\|\mathbf{C}^t \cdot \mathbf{x}_{i, \lambda} - \mathbf{m}^t\|}{\|\mathbf{C}^t \cdot (\mathbf{x}_{i, \lambda} - \mathbf{m}^t)\|} \times \frac{\mathbf{x}_{i, \lambda} - \mathbf{m}^t}{\sigma^t}
\]

\[
\sigma^{t+1} = \sigma^t \times \exp \left( \frac{\max_{\mathbf{p}_m^{t+1}, \mathbf{p}_c^{t+1}} \left( \frac{\|\mathbf{p}_m^{t+1}\|}{\|\mathbf{N}(0, \mathbf{I})\|} - 1 \right) \right) \quad \text{for } \|\mathbf{N}(0, \mathbf{I})\| = \sqrt{2 \Gamma(\frac{D+1}{2})/\Gamma(\frac{D}{2})} \text{ we use the approximation } \sqrt{D} \left(1 - \frac{1}{1 + \frac{1}{1 + \frac{D}{4}}}ight)
\]

Figure 1: Update equations for the state variables in the \((\mu/\mu_w, \lambda)\)-aCMA-ES with iteration index \( t = 0, 1, 2, \ldots \) and \( a \land bc + d = \min(a, bc + d) \). The chosen ordering of equations allows to remove the time index in all variables but \( \mathbf{m}^t \). The symbol \( \mathbf{x}_{i, \lambda} \) is the \( i \)-th best of the solutions \( \mathbf{x}_1, \ldots, \mathbf{x}_\lambda \).

real-valued evolutionary optimization. Partly due to the fact that the algorithm is quasi-parameter-free, the CMA-ES is widely applied and has become a quasi-standard.

A Further Improvement. The active CMA-ES proposed in [14] introduces a negative update of the covariance matrix in the \((\mu/\mu_t, \lambda)\)-CMA-ES. The authors investigate essentially unimodal functions. They observe a significant speed-up in particular on the discus function. The speed-up reaches almost a factor of three in dimension 20 (compared to the \((\mu/\mu_t, \lambda)\)-CMA-ES) and it increases with increasing dimension (i.e. the update leads to an improved scaling with increasing search space dimension). The negative update can in particular speed-up the adaptation of small variances in a small number of directions. The speed-up is less pronounced with a larger population size \( \lambda \).

This Paper. We combine the technique of negative updates with the \((\mu/\mu_w, \lambda)\)-CMA-ES and implement a weighted negative update procedure for the covariance matrix. The update also improves compared to [14], in that it remains feasible with a large population size. Consequently, the objective of this paper is threefold. 1) Specify a weighted negative update scheme with a parameter setting suited also for a very large population size. 2) Reproduce the effects from [14], observing significant improvements on functions, where a small number of directions need a small variance (here, we want to improve over the \((\mu/\mu_w, \lambda)\)-CMA-ES). 3) Survey the performance of a negative update on a larger number of diverse functions to possibly detect failures of the method. For this survey the BBOB-2010\(^4\) test environment is used.

2. THE ALGORITHM

The \((\mu/\mu_w, \lambda)\)-aCMA-ES is summarized in Figure 1. The initial values \( \mathbf{m}^{t=0} \in \mathbb{R}^D \) and \( \sigma^{t=0} > 0 \) are user defined and given in the next section. Modifications compared to [10, 6] are the addition of (7) and otherwise highlighted in pink (mainly the supplement to (8)).

The default parameter values are shown in Table 1. For \( c^{-} = 0 \), the \((\mu/\mu_w, \lambda)\)-CMA-ES is recovered.

Restarts. We apply nine restarts each with maximal 100 + 50(D + 3)^2/\sqrt{D} iterations. The default termination methods are used, besides that we have set \( \text{TolHistFun}=1e-12 \), \( \text{TolX}=2e-12 \), \( \text{StopOnStagnation}^{}=\text{on} \) and \( \text{MaxFunEvals}=\text{inf} \), following [6]. The population size \( \lambda \) is doubled for each restart, the first value is given in Table 1.

3. PARAMETER TUNING AND SETUP

The new parameter(s) for aCMA have been identified with experiments on the sphere function \( f_1 \) with various initial covariance matrices. In particular \( c^{-} \) is chosen such that (a) decreasing \( c^{-} \) from the default value does not improve the

\(^4\)http://coco.gforge.inria.fr/doku.php?id=bbob-2010
and processor (2.66 GHz) with Linux 2.6.28-18 and Matlab R2008a. In exploratory experiments comparing both
function, 15 trials are executed (on the first 15 instances)
while the best function value did not reach
stopping criterion for step-size
cumulation constant for step-size

cumulation constant for rank one update using $p_c$
covariance matrix learning rate for rank-µ update

Remark that $c^-$ becomes small, if $c^-$ gets close to one

with $\alpha_{\min} = \infty$, $\lambda_{\text{initarget}} = 0.66$ and $\lambda_{\max}(\cdot)$ is the largest eigenvalue; the

coefficient $c_{\min}$ depends in general on the iteration index, but here, due to the setting of $\alpha_{\min}$, no upper bound on $c^-$ is enforced

is in $[0, 1]$ and is chosen in the domain middle without deep motives, because it seems quite irrelevant

might become 1 in near future, which has only a negligible effect under neutral selection, because the term to the right of the $\wedge$ in (9) is approximately $\frac{1}{\alpha_{\sigma}} N(0, 1/2D)$ under neutral selection

CPU Timing Experiment. The complete algorithm was run on $f_9$ for at least 30 seconds on a Intel Core 2 6700 processor (2.66 GHz) with Linux 2.6.28-18 and Matlab R2008a. Results for the IPO-PACMA-ES are 2.0; 1.7; 1.4; 1.2; 1.1; 1.0; 0.8; 0.4 and 3.6 × 10⁻⁴ seconds per function evaluation for dimension 2; 3; 5; 10; 20; 40 and 80.

4. RESULTS

We show results for two algorithms: the IPO-PACMA-ES as presented above and denoted as aCMA, and the IPO-
CMA-ES, the same algorithm where $c^-$ is set to zero, de-
noted as CMA. In exploratory experiments comparing both
($\mu/\mu_\omega$, $\lambda$)-CMA-ES and ($\mu/\mu_\omega$, $\lambda$)-aCMA-ES with the result of the ($\mu/\mu_1$, $\lambda$)-variants from [14], the ($\mu/\mu_\omega$, $\lambda$)-variants perform at least as good and usually better: as a rule, ($\mu/\mu_\omega$, $\lambda$)-aCMA-ES outperforms ($\mu/\mu_1$, $\lambda$)-aCMA-ES that outperforms ($\mu/\mu_\omega$, $\lambda$)-CMA-ES.

Runtime results comparing aCMA with CMA and with the respective best algorithm from BBOB-2009 are presented

in Figures 2–5 and in Table 2. The expected running time (ERT), used in the figures and table, depends on a given target function value, $f_t = f_{\text{opt}} + \Delta f$, and is computed over all relevant trials as the number of function evaluations executed during each trial while the best function value did not reach $f_t$, summed over all trials and divided by the number of trials that actually reached $f_t$ [7, 16]. Statistical significance is tested with the rank-sum test for a given target $\Delta f_t (10^{-8}$ in Figure 5) up to the smallest number of function evaluations, $b_{\min}^e$, executed in any unsuccessful trial under consideration. The datum from each trial is either the best achieved $\Delta f$-value, or if $f_t$ was reached within the budget $b_{\min}^e$, the number of needed function evaluations to reach $\Delta f_t$ (inverted and multiplied by $-1$).

In the following, when a performance difference is highlighted on an individual function, the difference is statistically significant.

Figure 2. The figure shows Box-Whisker plots of the ERT ratio compared to the respective best algorithm from BBOB-

Table 1: Default parameter values of ($\mu/\mu_\omega$, $\lambda$)-aCMA-ES, where by definition $\sum_{i=1}^{\mu} w_i = 1$ and $\mu_\omega^3 = \sum_{i=1}^{\mu} w_i^2$ and $a - b \wedge c + d := \min(a - b, c + d)$. Only population size $\lambda$ is possibly left to the users choice (see also [1, 6])

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\lambda$</td>
<td>$4 + [3 \ln D]$</td>
</tr>
<tr>
<td>$\mu$</td>
<td>$\lfloor \frac{1}{\lambda} \rfloor$</td>
</tr>
<tr>
<td>$w_i$</td>
<td>$\frac{\ln (\frac{\lambda + 1}{\lambda}) - \ln i}{\sum_{j=1}^{\mu} \ln (\frac{\lambda + 1}{\lambda}) - \ln j}$</td>
</tr>
<tr>
<td>$c_{\min}$</td>
<td>1</td>
</tr>
<tr>
<td>$c_{\sigma}$</td>
<td>$\frac{\mu + 4}{\mu + 3}$</td>
</tr>
<tr>
<td>$d_{\sigma}$</td>
<td>$1 + c_{\sigma} + \max \left(0, \sqrt{\frac{\mu_{\omega} - 1}{\mu_{\omega} + 1}} - 1 \right)$</td>
</tr>
<tr>
<td>$c_c$</td>
<td>$\frac{\alpha_{\min} \min(1, \lambda/6)}{(D + 1)\lambda^{\alpha_{\min}}} + \frac{\mu + 1}{\mu_{\omega} + 1}$</td>
</tr>
<tr>
<td>$c_{\mu}$</td>
<td>$1 - c_{\lambda} - \alpha_{\min} (\frac{\mu_{\omega} - 2 + 1/\mu_{\omega}}{\mu_{\omega} + 1})$</td>
</tr>
<tr>
<td>$c_{\min}^-$</td>
<td>$c_{\min} - \min \left(1 - c_{\mu} \frac{\alpha_{\min}}{(D + 1)(\lambda^{\alpha_{\min}} + 1)} \right)$</td>
</tr>
<tr>
<td>$\alpha_{\sigma}^{-1}$</td>
<td>2</td>
</tr>
<tr>
<td>$\Delta_{\sigma}$</td>
<td>0.5</td>
</tr>
</tbody>
</table>

http://cocoa.gforge.inria.fr/doku.php?id=bbob-2010-results
2009 for all functions. Both algorithms show a very similar characteristic. Besides for very small budgets, the median (horizontal red line) and average (connected line) ERT ratio are somewhat below ten. In the final stage, aCMA improves in about 25% of the functions compared to the best algorithm from 2009 (ratio smaller than one).

**Figure 3.** The figure shows empirical cumulative distributions (a) of the runtime in number of function evaluations and (b) of the runtime ratio between the two algorithms aCMA/CMA. Both algorithms perform very similar while aCMA appears to be slightly faster. Clearly, the strongest effect in Fig. 3 is observed for the ill-conditioned functions in dimension 20. The runtime is consistently shorter for aCMA, almost uniformly by a factor of close to two. The improvement is statistically significant on all functions (see Table 2).

On the weak structure functions, the observed differences are probably caused by stochastic deviations and not significant. The remaining subgroups show almost identical behavior.

**Figure 4.** The scatter plots in Fig. 4 visualize the ratio of expected runtime aCMA/CMA for each measurement on each function and each dimension. A slightly improved scaling with aCMA can be observed on \( f_{22} \), \( f_{10} \) and \( f_{11} \). On the Discus function \( f_1 \) the effect is most pronounced and the speed-up comes close to a factor of three in dimension 40.

The advantage of CMA on \( f_{23} \) in dimension 10 is far beyond the maximum number of function evaluations, based on two successes (see Fig. 5) and not statistically significant.

Additionally, on functions \( f_7 \), \( f_{12} \)–\( f_{14} \), and \( f_{18} \) an advantage by aCMA, in particular for larger dimension, could be conjectured. On \( f_{10} \) an advantage of CMA could be conjectured. According to Table 2, in dimension 20, statistical significance is established for \( f_7 \) and \( f_{12} \)–\( f_{14} \). No such advantage is detected for CMA.

**Table 2.** Table 2 presents the ERT numbers for dimension 5 and 20 in comparison with the respective best algorithm of BBOB-2009. Again, aCMA is significantly better than CMA on \( f_2 \) and \( f_7 \)–\( f_{14} \) in dimension 20 as well as in dimension 5 apart from \( f_7 \) and \( f_{12} \).

Compared to the best algorithm from BBOB-2009, chosen for each function respectively, the aCMA can improve the record in dimension 20 on \( f_{10} \), \( f_{11} \), \( f_{14} \), \( f_{15} \) and \( f_{19} \). On further four functions aCMA is visibly better, but the results appear not to be statistically significant. The CMA improves the record on \( f_{15} \) and \( f_{19} \).

5. SUMMARY AND CONCLUSION

We have introduced the weighted negative update of the covariance matrix into the \((\mu/\mu_\alpha, \lambda)\)-CMA-ES with weighted recombination, denoted as \((\mu/\mu_\alpha, \lambda)\)-aCMA-ES. The negative update is based in the idea of [14]. It provides a similar effect for the \((\mu/\mu_\alpha, \lambda)\)-aCMA-ES as [14] for the \((\mu/\mu_\alpha, \lambda)\)-CMA-ES. The benchmarking of the new IPOP-aCMA-ES with BBOB-2010 reveals no deficiencies or failures compared to the IPOP-CMA-ES on the testbed of 24 functions in any dimension between 2 and 40.

The most prominent effect from aCMA is observed on ill-conditioned functions. In dimension 20, the average runtime advantage on the ill-conditioned functions is about 1.7 (CMA is 1.7 times slower than aCMA). On three ill-conditioned functions the scaling behavior compared to the CMA improves notably. In no case CMA outperformed aCMA with statistical significance.

Overall, the aCMA improves the performance over CMA on nine out of 12 essentially unimodal functions significantly at least in dimension 20, and the advantages are more pronounced in larger dimension. Finally, IPOP-aCMA-ES improves the record on five functions, where it is faster than the respective best algorithm from BBOB-2009 in dimension 20 with a sufficient statistical significance. In two of these cases IPOP-CMA-ES is still slightly faster and the record cannot be attributed to the negative covariance matrix update.

Acknowledgments

NH likes to thank M. Schoenauer for his support of BBOB.

6. REFERENCES


Figure 3: Empirical cumulative distributions (ECDF) of run lengths and speed-up ratios in 5-D (left) and 20-D (right). Left sub-columns: ECDF of the number of function evaluations divided by dimension to reach a target value $f_{\text{opt}} + \Delta f$ with $\Delta f = 10^k$, where $k \in \{-1, -4, -8\}$ is given by the first value in the legend, for IPOP-aCMA (solid) and IPOP-CMA (dashed). Light beige lines show the ECDF of FEvals for all functions that were solved in at least one trial (IPOP-aCMA first).
Figure 4: Expected running time (ERT in log10 of number of function evaluations) of IPOP-aCMA versus IPOP-CMA for 46 target values $\Delta f \in [10^{-8}, 10]$ in each dimension for functions $f_1$–$f_{24}$. Markers on the upper or right edge indicate that the target value was never reached by IPOP-aCMA or IPOP-CMA respectively. Markers represent dimension: 2: $+$, 3: $\triangledown$, 5: $\star$, 10: $\circ$, 20: $\Box$, 40: $\diamond$. Lines indicate the maximum number of function evaluations.
Figure 5: ERT ratio of IPOP-aCMA divided by IPOP-CMA versus $\log_{10}(\Delta f)$ for $f_1$-$f_{24}$ in 2, 3, 5, 10, 20, 40-D. Ratios $<10^0$ indicate an advantage of IPOP-aCMA, smaller values are always better. The line gets dashed when for any algorithm the ERT exceeds thrice the median of the trial-wise overall number of $f$-evaluations for the same algorithm on this function. Symbols indicate the best achieved $\Delta f$-value of one algorithm (ERT gets undefined to the right). The dashed line continues as the fraction of successful trials of the other algorithm, where 0 means 0% and the y-axis limits mean 100%, values below zero for IPOP-aCMA. The line ends when no algorithm reaches $\Delta f$ anymore. The number of successful trials is given, only if it was in $\{1 \ldots 9\}$ for IPOP-aCMA (1st number) and non-zero for IPOP-CMA (2nd number). Results are significant with $p = 0.05$ for one star and $p = 10^{-#*}$ otherwise, with Bonferroni correction within each figure.
Table 2: Expected running time (ERT in number of function evaluations) divided by the best ERT measured during BBOB-2009 (given in the respective first row) for different Δf values for functions f₁⁻f₂. The median number of conducted function evaluations is additionally given in italics, if ERT(10⁻⁷) = ∞. Bold entries are statistically significantly better compared to the other algorithm, with p = 0.05 or p = 10⁻k where k > 1 is the number following the * symbol, with Bonferroni correction of 48.

<table>
<thead>
<tr>
<th>Δf</th>
<th>1e+1</th>
<th>1e+0</th>
<th>1e-1</th>
<th>1e-2</th>
<th>1e-3</th>
<th>1e-4</th>
<th>Succe</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1.30</td>
<td>1.78</td>
<td>2.27</td>
<td>2.75</td>
<td>3.23</td>
<td>3.71</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>2.34</td>
<td>3.02</td>
<td>3.71</td>
<td>4.40</td>
<td>5.10</td>
<td>5.80</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>2.34</td>
<td>3.02</td>
<td>3.71</td>
<td>4.40</td>
<td>5.10</td>
<td>5.80</td>
<td></td>
</tr>
</tbody>
</table>

5-D

<table>
<thead>
<tr>
<th>Δf</th>
<th>1e+1</th>
<th>1e+0</th>
<th>1e-1</th>
<th>1e-2</th>
<th>1e-3</th>
<th>1e-4</th>
<th>Succe</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2.70</td>
<td>3.56</td>
<td>4.52</td>
<td>5.50</td>
<td>6.50</td>
<td>7.50</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1.30</td>
<td>1.78</td>
<td>2.27</td>
<td>2.75</td>
<td>3.23</td>
<td>3.71</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>1.30</td>
<td>1.78</td>
<td>2.27</td>
<td>2.75</td>
<td>3.23</td>
<td>3.71</td>
<td></td>
</tr>
</tbody>
</table>

20-D

<table>
<thead>
<tr>
<th>Δf</th>
<th>1e+1</th>
<th>1e+0</th>
<th>1e-1</th>
<th>1e-2</th>
<th>1e-3</th>
<th>1e-4</th>
<th>Succe</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2.70</td>
<td>3.56</td>
<td>4.52</td>
<td>5.50</td>
<td>6.50</td>
<td>7.50</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1.30</td>
<td>1.78</td>
<td>2.27</td>
<td>2.75</td>
<td>3.23</td>
<td>3.71</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>1.30</td>
<td>1.78</td>
<td>2.27</td>
<td>2.75</td>
<td>3.23</td>
<td>3.71</td>
<td></td>
</tr>
</tbody>
</table>

---

**Reference:**


