A comparative introduction to two optimization classics: the Nelder-Mead and the CMA-ES algorithms
Rodolphe Le Riche

To cite this version:
Rodolphe Le Riche. A comparative introduction to two optimization classics: the Nelder-Mead and the CMA-ES algorithms. Doctoral. La Rochelle, France. 2018. cel-02091527

HAL Id: cel-02091527
https://hal.archives-ouvertes.fr/cel-02091527
Submitted on 5 Apr 2019

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.
A comparative introduction to two optimization classics: the Nelder-Mead and the CMA-ES algorithms

Rodolphe Le Riche\textsuperscript{1,2}

\textsuperscript{1} Ecole des Mines de Saint Etienne, France
\textsuperscript{2} CNRS LIMOS France

March 2018
MEXICO network research school on sensitivity analysis, metamodeling and optimization of complex models
La Rochelle, France
In this 1h30 course + 1h30 hands-on class, we address unconstrained optimization with continuous variables

\[
\begin{align*}
\text{find one or many} & \quad x^* \equiv \arg \min_{x \in S \in \mathbb{R}^d} f(x) \\
\end{align*}
\]

where \(S\) is typically the compact \([x_{\text{LB}}, x_{\text{UB}}] \in \mathbb{R}^d\).

We cover the principles (the optimization “engines”) of two algorithms, Nelder-Mead and CMA-ES. These algorithms have the same goal, but one is deterministic and one is stochastic.

Both algorithms do not use metamodels: because they do not model \(f\), they are insensitive to a monotonic transformation of \(f()\) (e.g., they perform the same on \(f()\), \(f^3()\), \(\exp(f())\)); they do not build any approximation of \(f()\) valid in the entire \(S\) (like EGO does).
Optimizers are iterative algorithms

We are going to approximate the solution(s) to Problem (1) using iterative optimization algorithms (optimizers) that exchange information with the objective function code (that wraps the often costly simulation).

\[
\text{optimizer} \quad \min_x f(x) \quad \text{state } S
\]

or in a state-space representation,

\[
x^{t+1} = \phi(S^{t+1}) \quad (2)
\]
\[
S^{t+1} = \psi(S^t, (x^t, f(x^t))) \quad (3)
\]

The optimizer is defined by \( S \) (state vector), \( \phi \) (next iterate equation) and \( \psi \) (state equation).
Content of the talk

- A pattern search method: the Nelder-Mead algorithm
- A stochastic optimizer: CMA-ES
The Nelder-Mead algorithm


It belongs to the class of pattern search methods [2], i.e., methods that propose new points to evaluate based on the deformation of a geometrical pattern.

Sometimes also called “simplex” (not to be mistaken with that of linear programming) because the pattern is made of a simplex: \( d + 1 \) points in a \( d \)-dimensional space.
Simplex properties (1)

Notation: \( f(x^0) \leq f(x^1) \leq \ldots \leq f(x^d) \)

\((x^1 - x^0, x^2 - x^0)\) is a basis of \( \mathbb{R}^d \).

\((x^1 - x^0, x^2 - x^0, -(x^1 - x^0), -(x^2 - x^0))\) a positive set \((-\nabla f(x^0)\) can be approximated as a positive linear combination of the vectors, a condition for convergence with shrink steps, see later).

\(d + 1\) is the minimum number of points to construct a linear interpolation.
Simplex properties (2)

$d + 1$ is the minimum number of points to construct a linear interpolation:

$$
\hat{f}(\mathbf{x}^i) = \alpha_0 + \alpha_1 x_1^i + \ldots + \alpha_d x_d^i = f(\mathbf{x}^i), \quad i = 0, d
$$

$$
\begin{bmatrix}
1 & x_1^0 & \ldots & x_d^0 \\
1 & x_1^1 & \ldots & x_d^1 \\
\vdots & & \ddots & \vdots \\
1 & x_1^d & \ldots & x_d^d
\end{bmatrix}
\begin{pmatrix}
\alpha_0 \\
\alpha_1 \\
\vdots \\
\alpha_d
\end{pmatrix}
= 
\begin{pmatrix}
f(\mathbf{x}^0) \\
f(\mathbf{x}^1) \\
\vdots \\
f(\mathbf{x}^d)
\end{pmatrix}
$$

$$
\Leftrightarrow
\begin{bmatrix}
1 & x_1^0 & \ldots & x_d^0 \\
0 & x_1^1 - x_1^0 & \ldots & x_d^1 - x_0^0 \\
\vdots & & \ddots & \vdots \\
0 & x_1^d - x_1^0 & \ldots & x_d^d - x_d^0
\end{bmatrix}
\begin{pmatrix}
\alpha_0 \\
\alpha_1 \\
\vdots \\
\alpha_d
\end{pmatrix}
= 
\begin{pmatrix}
f(\mathbf{x}^0) \\
f(\mathbf{x}^1) - f(\mathbf{x}^0) \\
\vdots \\
f(\mathbf{x}^d) - f(\mathbf{x}^0)
\end{pmatrix}
$$

which has a unique solution iff lower right sub-matrix $\mathbf{L}$ is invertible, or better, well-conditioned.
$d + 1$ is the minimum number of points to construct a linear interpolation:

$$\hat{f}(x^i) = \alpha_0 + \alpha_1 x_1^i + \ldots + \alpha_d x_d^i = f(x^i), \quad i = 0, d$$

$$\begin{bmatrix}
1 & x_0^0 & \ldots & x_d^0 \\
1 & x_1^0 & \ldots & x_d^0 \\
\vdots & \vdots & \ddots & \vdots \\
1 & x_1^d & \ldots & x_d^d \\
\end{bmatrix} \begin{pmatrix}
\alpha_0 \\
\alpha_1 \\
\alpha_2 \\
\end{pmatrix} = \begin{pmatrix}
f(x_0^0) \\
f(x_1^0) \\
\vdots \\
f(x_d^0) \\
\end{pmatrix}$$

$$\Leftrightarrow \begin{bmatrix}
1 & x_0^0 & \ldots & x_d^0 \\
0 & x_1^0 - x_0^0 & \ldots & x_d^0 - x_0^0 \\
0 & x_1^d - x_0^0 & \ldots & x_d^d - x_0^0 \\
\end{bmatrix} \begin{pmatrix}
\alpha_0 \\
\alpha_1 \\
\alpha_2 \\
\end{pmatrix} = \begin{pmatrix}
f(x_0^0) \\
f(x_1^0) - f(x_0^0) \\
\vdots \\
f(x_d^0) - f(x_0^0) \\
\end{pmatrix}$$

which has a unique solution iff lower right sub-matrix $L$ is invertible, or better, well-conditioned.
The diameter of a simplex \( \{ x^0, \ldots, x^d \} \) is its largest edge,

\[
\text{diam}(\{ x^0, \ldots, x^d \}) = \max_{0 \leq i < j \leq d} \| x^i - x^j \|
\]

A measure of how well the edges of a simplex span \( \mathbb{R}^d \) is its normalized volume (noting \( s \equiv \text{diam}(\{ x^0, \ldots, x^d \}) \)),

\[
\overline{\text{vol}}(\{ x^0, \ldots, x^d \}) = \text{vol}\left(\frac{x^1 - x^0}{s}, \ldots, \frac{x^d - x^d}{s}\right) = \frac{\det(L)}{d!s^d},
\]

which is a scale invariant measure of the volume of the simplex.
A Nelder-Mead iteration

New points are generated through simple geometrical transformations. Remember $f(x^0) \leq f(x^1) \leq \ldots \leq f(x^d)$. 
**A Nelder-Mead iteration**

**Reflection:** take a step from the worst to its symmetric w.r.t. the centroid of the $d$ better points $\approx$ a descent step, whose size and direction come from the current simplex,

$$x^r = x^c + \delta_r (x^c - x^d)$$

where $\delta_r = 1$ and $x^c = \frac{1}{d} \sum_{i=0}^{d-1} x^i$

calculate $f(x^r)$ (cost 1 call to $f$).

If $f(x^0) \leq f(x^r) < f(x^{d-1})$, $x^d \leftarrow x^r$, end iteration.
A Nelder-Mead iteration

**Expansion:** if $f(x^r) < f(x^0)$, progress was made, increase step size.

\[ x^e = x^c + \delta_e (x^c - x^d) \quad \text{where} \quad \delta_e = 2 \]

If $f(x^e) \leq f(x^r)$, $x^d \leftarrow x^e$, end iteration.
Otherwise, $x^d \leftarrow x^r$, end iteration.
A Nelder-Mead iteration

**Contraction:** if \( f(x^r) \geq f(x^{d-1}) \), little progress, reduce step size: Outside contraction if \( f(x^r) < f(x^d) \); Inside contraction if \( f(x^r) \geq f(x^d) \).

\[
x^{c\{i,o\}} = x^c + \delta_{c\{i,o\}}(x^c - x^d)
\]
where \( \delta_{ci} = -1/2 \) and \( \delta_{co} = 1/2 \)

If \( f(x^{co}) \leq f(x^r) \), \( x^d \leftarrow x^{co} \); Else \( x^d \leftarrow x^r \); End iteration.

If \( f(x^{ci}) < f(x^d) \), \( x^d \leftarrow x^{ci} \) and end iteration; Else shrink.
A Nelder-Mead iteration

**Shrink:** if \( f(x^c_i) > f(x^d) \), no progress, reduce simplex size (direction more influenced by \( x^0 \) and smaller step size).

\[
x^{s\{1,...,d\}} = x^0 + \gamma_s (x^{\{1,...,d\}} - x^0)
\]

where \( \gamma_s = 1/2 \)

Evaluate \( f \) at \( x^{s1}, \ldots, x^{sd} \) (cost \( d \)), end iteration.
Flow chart of the Nelder-Mead algorithm

1. Initialize simplex, min diam, max calls (to \( f \))
2. Do a Nelder Mead iteration (and reorder points with increasing \( f \))
3. Check if \( \text{diam(simplex)} < \text{min diam} \) or \( \text{nb calls to } f > \text{max calls} \)?
4. If yes: Stop
5. If no: Go back to step 2
Properties of the Nelder-Mead algorithm

- An iteration may cost 1 call to $f$ if end at reflection, or 2 calls (reflection + expansion; reflection + contraction) or $d + 2$ calls (reflection + expansion + shrink).
- It always converges to a stationary point (where $\nabla f(x) = 0$) when $d = 1$ (proof in [8]) and $f$ is continuously differentiable.
- However it may converge to nonstationary points:

Nelder-Mead operates a series of contractions and converges to a nonstationary point on McKinnon functions which are convex and continuously differentiable (plot from [12]).
Practicalities

- **Re_starts**: after each iteration, add a restart test, if $\text{vol} < \varepsilon$ (say $10^{-4}$), restart the simplex at $x^0$.

- **Bound constraints handling**, $x^{LB} \leq x \leq x^{UB}$: the solution used in the nlopt software ([6], a R version exists) is to project points onto the constraints whenever a Nelder Mead transformation creates out-of-bounds points. This may induce a collapse of the simplex onto a subspace and prevent convergence.

- Works often well when $d \leq 10$ even on discontinuous, moderately multi-modal, moderately noisy functions. Nelder-Mead seems to contain a minimal set of operations to rapidly adapt to $f$ local curvatures.
Sample runs on two identification test cases, where the minimization problem has the form

1. a simplified material law identification from a tension test
2. the identification of
Converging extensions to the Nelder-Mead algorithm

- Do a $180^\circ$ rotation of the simplex when the reflection does not work ([2, 16]): ensures that a descent direction can be found if the simplex is small enough (either $d^\top \nabla f(x^0) < 0$ or $-d^\top \nabla f(x^0) < 0$, unless at a stationary point).
- ... (393 articles with Nelder Mead in the title on Google Scholar).
A stochastic way of searching for good points. Replace the previous (simplex / simplex transformations) by (multivariate normal distribution / sampling and estimation).

\( \mathcal{N}(\mathbf{m}, \mathbf{C}) \) represent our belief about the location of the optimum \( \mathbf{x}^* \).
A $d$-dimensional Gaussian is fully described by its mean and covariance $\Rightarrow$ the central question is “how to adapt the mean and covariance of $\mathcal{N}$ to optimize efficiently?”.

Sample generation:

- $L \leftarrow \text{cholesky}(C)$
- $U \sim \mathcal{N}(0, I)$ \{rnorm in R\}
- $x \leftarrow m + L \times U$

because if $\text{Cov}(U) = I$ and $x = Lu$, then $\text{Cov}(X) = L\text{Cov}(U)L^\top = LL^\top = C$
Originated in the evolutionary computation community with the work of Niko Hansen et al. in 1996 [4, 5], now turning more mathematical (cf. work by Anne Auger, [1, 15], Dimo Brockhoff, …).

A practical tutorial is [3].

The presentation is restricted to the main concepts. To start with, ES-(1+1) with constant step.
The constant step size ES-(1+1)

Algorithm 1 constant circular step ES-(1+1)

Require: \( m \in S, \sigma^2, t^{\text{max}} \)
1: calculate \( f(m) \), \( t \leftarrow 1 \)
2: while \( t < t^{\text{max}} \) do
3: sample: \( x \sim \mathcal{N}(m, \sigma^2 I) \)
4: calculate \( f(x) \), \( t \leftarrow t + 1 \)
5: if \( f(x) < f(m) \) then
6: \( m \leftarrow x \)
7: end if
8: end while

The simplest stochastic optimizer: the covariance matrix is \( C = \sigma^2 I \).
The choice of $\sigma^2$ is critical

Minimize the sphere function $f(x) = \|x\|^2$, $x \in \mathbb{R}^{10}$, starting from $m = (4, \ldots, 4)^\top$ with ES-(1+1) constant circular step size, 10 repetitions. Plot shows the 25%, 50% and 75% quantiles.

A constant step size may go from too small to too large during convergence.

The optimal step size for the sphere function is (proof in a later slide)

$$\sigma^* \approx 1.22 \frac{\|x\|}{d}$$
In the simple ES-(1+1), the perturbation seen by the current mean $\mathbf{m}$ is $\|\mathbf{x} - \mathbf{m}\|^2 = \sigma^2 \sum_{i=1}^{d} u_i^2$ where $u_i \sim \mathcal{N}(0, 1)$ iid, $\|\mathbf{x} - \mathbf{m}\|^2 \sim \sigma^2 \chi^2_d$.

As $d \nearrow$, $\chi^2_d \to \mathcal{N}(d, 2d)$ and $\sqrt{\chi^2_d} \to \mathcal{N}(\sqrt{d}, 1/2)$ (see graph, $d = 2, 10, 100$)

$\Rightarrow$ as $d \nearrow$, the points are concentrated in a sphere of radius $\sigma \sqrt{d}$ centered at $\mathbf{m}$. 

$\mathbf{m}$ is the current mean.

$$
\|\mathbf{x} - \mathbf{m}\|^2 = \sigma^2 \sum_{i=1}^{d} u_i^2
$$

$\mathbf{m}$ is the current mean.
Optimal step size of ES-(1+1) with the sphere function (1)

The progress into $f$ is
$$P_2 = \|m\|^2 - \|m + \sigma U\|^2$$
$$= m^\top m - (m + \sigma U)^\top (m + \sigma U)$$
$$= -\sigma^2 U^\top U - 2\sigma m^\top U$$

$P_2 \approx -\sigma^2 d - 2\sigma m^\top U$

The expected progress of ES-(1+1) is
$$\mathbb{E} [\max(0, P_2)] = \int_0^\infty P_2 p(P_2) dP_2 = \int_{\sigma d/(2\|m\|)}^{+\infty} (2\sigma \|m\| p - \sigma^2 d) \phi(p) dp$$

with the normalization $p = (P_2 + \sigma^2 d)/(2\sigma \|m\|)$ and $\phi()$ pdf of $i \sim \mathcal{N}(0, 1)$

The expected progress of ES-(1+1) is
$$\mathbb{E} [\max(0, P_2)] = \int_0^\infty P_2 p(P_2) dP_2 = \int_{\sigma d/(2\|m\|)}^{+\infty} (2\sigma \|m\| p - \sigma^2 d) \phi(p) dp$$

with the normalization $p = (P_2 + \sigma^2 d)/(2\sigma \|m\|)$ and $\phi()$ pdf of $i \sim \mathcal{N}(0, 1)$
Optimal step size of ES-(1+1) with the sphere function (2)

\[ \mathbb{E} \left[ \max(0, P_2) \right] = -\sigma^2 d \left[ 1 - \Phi \left( \frac{\sigma d}{2 \|m\|} \right) \right] + 2\sigma \|m\| \phi \left( \frac{\sigma d}{2 \|m\|} \right) \]

where \( \Phi() \) denotes the cdf of \( \mathcal{N}(0, 1) \). Multiplying by \( \frac{d}{\|m\|^2} \) and introducing the normalized step \( \bar{\sigma} = \frac{\sigma d}{\|m\|} \), the normalized progress is

\[ \frac{d}{\|m\|^2} \mathbb{E} \left[ \max(0, P_2) \right] = \sigma^2 \left[ \Phi \left( \frac{\bar{\sigma}}{2} \right) \right] + 2\sigma \phi \left( \frac{\bar{\sigma}}{2} \right) \]

which is maximized at \( \bar{\sigma}^* \approx 1.22 \Rightarrow \sigma^* \approx 1.22 \frac{\|m\|}{d} \)
Optimal step size of ES-(1+1) with a parabolic function

Let optimize $f(x) = \frac{1}{2} x^\top H x$ with an ES-(1+1) whose steps follow $x - m \sim \mathcal{N}(0, \sigma^2 \overline{C})$.

$\overline{C}$ can be decomposed into the product of its eigenvectors $B$ and eigenvalues $D^2$, $\overline{C} = BD^2B^\top$.

We choose $\overline{C} = H^{-1}$ and perform the change of variables $y = D^{-1}B^\top x$. Then, $f(x) = \frac{1}{2} x^\top BD^{-1}D^{-1}B^\top x = \frac{1}{2} y^\top y = f(y) \Rightarrow$ in the $y$-space $f$ is a sphere.

The covariance of $Y$ is $C_y = \sigma^2 D^{-1}B^\top BD^2B^\top BD^{-1} = \sigma^2 I$

$\Rightarrow$ sphere function and spherical perturbations, we can use the result

$\sigma^* = 1.22 \frac{\|y\|}{d} = \frac{1.22}{d} \sqrt{x^\top BD^{-2}B^\top x} = \frac{1.22}{d} \sqrt{x^\top H x}$

$C$ proportional to the inverse of the Hessian allows to use the sphere optimality results and is the best choice when there is only local information ($\nabla f(m), \nabla^2 f(m)$).

This is what CMA-ES learns.
ES-(\(\mu, \lambda\)) flow chart

CMA-ES is an evolution strategy ES-(\(\mu, \lambda\)): \(\lambda\) points sampled, the \(\mu\) best are used for updating \(m\) and \(C\).

Algorithm 2 Generic ES-(\(\mu, \lambda\))

Require: \(m \in \mathcal{S}, C, t^{\text{max}}\)

1: calculate \(f(m), t \leftarrow 1, g \leftarrow 1\)
2: while \(t < t^{\text{max}}\) do
3: sample: \(x^1, \ldots, x^\lambda \sim \mathcal{N}(m^{(g)}, C^{(g)})\)
4: calculate \(f(x^1), \ldots, f(x^\lambda), t \leftarrow t + \lambda\)
5: rank: \(f(x^{1:\lambda}) \leq \ldots \leq f(x^{\lambda:\lambda})\)
6: estimate \(m^{(g+1)}\) and \(C^{(g+1)}\) with the \(\mu\) bests, \(x^{1:\lambda}, x^{2:\lambda}, \ldots, x^{\lambda:\lambda}\).
7: \(g \leftarrow g + 1\)
8: end while

Default semi-empirical values for CMA-ES (N. Hansen):
\(\lambda = 4 + \lfloor 3 \ln(d) \rfloor\), \(\mu = \lfloor \frac{\lambda}{2} \rfloor\).
A simplified CMA-ES

We now present a simplification of the CMA-ES algorithm that is easier to understand while keeping some of the key ideas. It will not compare in performance with the true CMA-ES. Disregarded ingredients are

- weighting of the sampled points
- separation of the adaptation of the step size and covariance “shape”, \( \mathbf{C} = \sigma^2 \overline{\mathbf{C}} \)
- simultaneous rank 1 and rank \( \mu \) updates of the covariance matrix

For competitive implementations, refer to the official tutorial [3].
Points are generated from sampling, at iteration $g$, $\mathcal{N}(\mathbf{m}^{(g)}, \mathbf{C}^{(g)})$,

$$x^i = \mathbf{m}^{(g)} + \mathcal{N}_i(0, \mathbf{C}^{(g)}) \quad , \quad i = 1, \lambda$$

The associated step is

$$y^i = x^i - \mathbf{m}^{(g)} = \mathcal{N}_i(0, \mathbf{C}^{(g)})$$

CMA-ES learns steps, as opposed to points. Points are static, steps are dynamic. Efficient optimizers (e.g., Nelder Mead, BFGS, ...) make steps.
Points versus steps (2)

Note how the law of good steps stretches in the good direction as opposed to the good points (plot from [3])

\[ \mathcal{N} \text{ of good steps centered at } \mathbf{m}^{(g+1)} \]

\[ \mathcal{N} \text{ of good points centered at } \mathbf{m}^{(g+1)} \]
Update of the mean

Remember the order statistics notation for the new sampled points

\[ f(x^{1:λ}) \leq \ldots \leq f(x^{λ:λ}) \]

The new mean is the average of the μ best points,

\[ m^{(g+1)} = \frac{1}{μ} \sum_{i=1}^{μ} x^{i:λ} \]
Covariance matrix estimation

Empirical estimate of the good steps,

\[
C^{(g+1)} = \frac{1}{\mu} \sum_{i=1}^{\mu} (x_i^{i:}\lambda - m^{(g)})(x_i^{i:}\lambda - m^{(g)})^\top
\]

\[
= \frac{1}{\mu} \sum_{i=1}^{\mu} y_i^{i:}\lambda y_i^{i:}\lambda^\top
\]

Empirical estimate of the good points (don’t use),

\[
C_{EDA}^{(g+1)} = \frac{1}{\mu - 1} \sum_{i=1}^{\mu} (x_i^{i:}\lambda - m^{(g+1)})(x_i^{i:}\lambda - m^{(g+1)})^\top
\]


Both estimators are unbiased. They require \( \mu \geq d \) linearly independent \( y \)'s to have full rank and \( \lambda \geq 20d \) to be reliable for well-conditioned (\( \text{cond}(C) < 10 \)) problems [3].
If $f$ is ill-conditioned (large ratio of largest to smallest curvatures), the estimator (4) requires large $\mu$ (hence $\lambda$) which makes the optimizer costly.

Solution: account for information coming from past iterations by time averaging, which allows few samples and rank 1 updates.

Use the average good step,

$$
\bar{y} = \frac{1}{\mu} \sum_{i=1}^{\mu} (x^{(i:\lambda)} - m^{(g)}) = m^{(g+1)} - m^{(g)}
$$

(5)
Rank 1 and time averaging for covariance matrix estimation (2)

and time averaging of the covariance estimates,

\[ \mathbf{C}^{(g+1)} = (1 - c_1)\mathbf{C}^{(g)} + c_1\mathbf{y}\mathbf{y}^\top \]  

(6)

where \(0 \leq c_1 \leq 1\).

If \(c_1 = 1\), no memory and \(\mathbf{C}^{(g+1)}\) has rank 1 (degenerated gaussian along \(\mathbf{y}\)).

Default\(^1\) \(c_1 = 2/((d + 1.3)^2 + \mu)\).

\(1/c_1\) is the backward time horizon that contributes to about 63\% of the information [3].

\(^1\)Default for the complete CMA-ES, not sure it applies to this simplified version. Idem with \(c_c\) later.
Steps cumulation

The estimation (6) has a flaw: it is not sensitive to the sign of the step, \( \overline{y} \overline{y}^\top = (\overline{-y})(\overline{-y})^\top \). It will not detect oscillations and cannot converge. Worse, it will increase step size around \( x^* \).

\[ \Rightarrow \text{replace } \overline{y} \text{ from (6) by the cumulated steps (aka evolution path) } p \text{ below,} \]

\[ p^{(g+1)} = (1 - c_c)p^{(g)} + \sqrt{c_c(2 - c_c)\mu} \overline{y} \]  

(7)

Default \( c_c = \frac{4+\mu/d}{d+4+2\mu/d} \).

The weighting in (7) is such that if \( y^{i:\lambda} \sim N(0, C) \) and \( p^{(g)} \sim N(0, C) \), then \( p^{(g+1)} \sim N(0, C) \).

Proof: \( \overline{y} = \frac{1}{\mu} \sum_{i=1}^{\mu} y^{i:\lambda} \sim N(0, \frac{\mu}{\mu^2} C) \), \( p^{(g+1)} \sim N(0, (1 - c_c)^2 C) + N(0, c_c(2-c_c)/(\mu) C) \sim N(0, C) \). □
Practicalities

- Accounting for the variables bounds:

```plaintext
repeat
    sample: \( x \sim \mathcal{N}(m, C) \)
until \( x^{LB} \leq x \leq x^{UB} \)
```

- Another stopping criterion: a maximum number of \( f \) calculations without progress (to avoid stagnation).
Concluding remarks I

- Both algorithms are likely to miss the global optimum: practically, they are local (although CMA-ES is asymptotically convergent to the global optimum and Nelder-Mead may go over regions of attraction to local optima).

- Restart Nelder-Mead and CMA-ES at convergence (possibly force a few early convergences for restarts). Competitive versions of these algorithms use restarts (e.g., [9, 11]).
Concluding remarks II

- A tempting idea when \( f() \) is costly:
  1. Make a design of experiments \((\mathbf{X}, F)\) and build a metamodel \( \hat{f} \)
  2. Optimize the (almost free) \( \hat{f} \) with your favorite optimizer,
     \[ x^{t+1} = \arg \min_{x \in S} \hat{f}(x) \]
  3. Calculate \( f(x^{t+1}) \), stop or back to 1 with
     \((\mathbf{X}, F) \cup (x^{t+1}, f(x^{t+1}))\).

  but convergence may be harmed if \( \hat{f}(x^{t+1}) \approx f(x^{t+1}) \) and \( x^{t+1} \)
  far from \( x^* \) ⇒ use a well-designed optimizer for this, e.g., EGO [7], saACM-ES [10] or an EGO-CMA mix [13].

Acknowledgments: I would like to thank Victor Picheny, Stéphanie Mahévas and Robert Faivre for their invitation at this MEXICO network school.
References I

Linear convergence of comparison-based step-size adaptive randomized search via stability of markov chains.

*Introduction to Derivative-Free Optimization*.

The CMA Evolution Strategy: A Tutorial.

Adapting arbitrary normal mutation distributions in evolution strategies: The covariance matrix adaptation.

Completely derandomized self-adaptation in evolution strategies.


Convergence of the nelder–mead simplex method to a nonstationary point.

Making EGO and CMA-ES Complementary for Global Optimization.

A simplex method for function minimization.

Information-geometric optimization algorithms: A unifying picture via invariance principles.
accepted.

Fortified-descent simplicial search method: A general approach.