On the Comparison Between Derivative-Free and Gradients Methods for Constrained Optimization

IEOR 262B, University of California, Berkeley

(Report)

Jules Bertrand[1], Adam Moutonnet[2]
[1]jules.bertrand@berkeley.edu, [2]adam_moutonnet@berkeley.edu

I. Introduction

In a world where performance is the watchword in absolutely every field, from agriculture to finance, healthcare, aeronautics and many others, optimization has become essential. The principle of optimization is very easy. Tune a finite set of parameters in order to maximize or minimize a metric describing the performance of a model or a machine. The very essence of the research field that is optimization is to always find better ways of tuning these parameters to reach the summit faster.

Commonly, scientists define a function f over a set $X \subset \mathbb{R}^n$ with values in $\mathbb{R}$, where X is the set of tunable parameters. The maximization (resp. minimization) of this function is very often realized thanks to a gradient ascent (resp. descent). It is possible thanks to the fact that gradient always points in the direction of ascent. Taking a step in its direction maximizes the value of the function. In the minimization case, one just have to take a step in the opposite direction of it. However, many functions have gradient that are very hard or costly to process, or does not even exist at some points. This problem might be overcome by the existence of sub-gradients [3], but even sub-gradients are not guaranteed to exist for non-smooth functions. Moreover, one can only optimize using gradients when the function form is known. Unfortunately in many complex problems such as machine learning we have to deal with *black-box* functions whose form is not known at all.

For all these reasons, scientists developed so called *derivative-free optimization* (DFO) methods. These methods aim at finding the optimum of a function without having to know the derivative of it or even its definition by simply evaluating points in the parameters set X, where $X \neq \mathbb{R}^n$. Our goal throughout this project is to explore two of them that are *Mesh Adaptive Direct Search* described by [2] and *Covariance Matrix Adaptation Evolution Strategy* described by [4] in its current form for unconstrained optimization and [1] in the case of constrained optimization with a single linear constraint. We will first give an overview of the mechanisms behind these two methods. Then, after implementing these two DFO methods we will assess their performance and those of some gradient-based methods, on unconstrained and constrained optimization problems for smooth, non-smooth, convex, non-convex, well or ill-conditioned functions. Finally, we will give a global comparison between DFO and gradient-based methods to highlight the advantages and the drawbacks of this or this method. From now on we will only focus on minimization problems without loss of generality:

$$\begin{aligned} \min_{x} \quad & f(x) \\ s.t. \quad & g(x) = Ax - b \leq 0 \quad \text{(for the constrained problem only)} \\ & x \in \mathbb{R}^n \end{aligned}$$

II. Description of the Methods

A. Mesh Adaptive Direct Search

A classic Mesh Adaptive Direct Search (MADS) method, as the name suggests, consists of evaluating some points belonging to a mesh around the current point of interest to find a point decreasing the function value. At step k of the method, we refer to as a mesh around our point of interest $x_k \in \mathbb{R}^n$ any set of points M_k defined such that

$$M_k = \{x_k + \lambda_k D z \mid z \in \mathbb{N}^{n_D}\}$$

Where $\lambda_k > 0$ is the mesh step size at step k, D is a matrix whose columns are positively spanning $\mathbb{R}^n$ (a non-negative linear combination of them must span $\mathbb{R}^n$) and n_D is the number of columns in D. This process will either lead to the discovery of a better point or to nothing. If a better point is discovered, i.e. a point decreasing the function value, it will become our new point of interest and we will increase the mesh step size. In the case where there are multiple better points, we can choose either the best one, or the first one discovered in an opportunistic approach which is less costly in evaluations. Otherwise, we keep the same point of interest and reduce the mesh step size. A property for the sequence $\{\lambda_k\}$ is then that $\lim_k \inf_{l \in \{0,\dots,k\}} \lambda_l = 0$. The MADS method is particularly useful on topological situations like plateaus, ridges, etc... where gradients lose their efficiency. Note that in practice the mesh does not need to be constructed in its totality before starting to evaluate points, one can easily construct these points one after another by ensuring that they all belong to the mesh. An illustration of this method is shown on Figure 1.

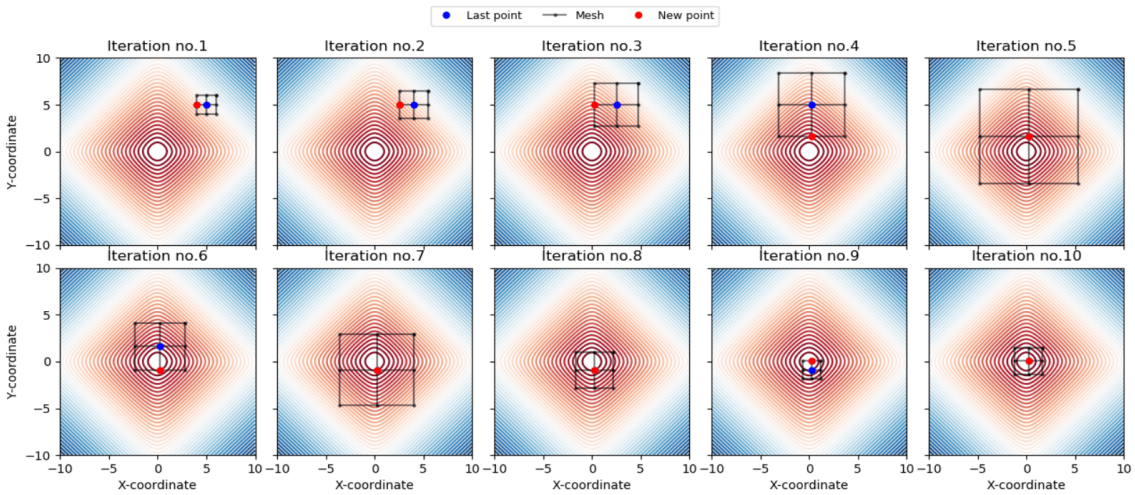

Fig. 1. Illustration of MADS search mechanism with opportunistic approach for $f : (x, y) \to \sqrt{1+x^2} + \sqrt{1+y^2}$ in $\mathbb{R}^2$. Red represents lower values of f. Here, D is the maximal positive canonical basis of $\mathbb{R}^2$, and z is drawn in $\{0, 1\}^4$.

In their paper [2], C. Audet and J.E. Denis propose to add another step in this process. A first step called the *search* step implements the classic MADS method. If no point is found after this first step, the *poll* step begins and consists of evaluating random points on a finer mesh called a *frame* within the first mesh. At step k of the algorithm, the frame P_k around the current point of interest x_k is such that

$$P_k = \{x_k + \mu_k d \mid d \in D_k\} \subset M_k$$

Where μ_k is the frame step size such that $\lambda_k \geq \mu_k > 0$ and D_k is a positive spanning set of $\mathbb{R}^n$ drawn randomly such that the closure of the cone generated by the set

$$\bigcup_{k=1}^{\infty} \left\{ \frac{d}{\|d\|} \mid d \in D_k \right\}$$

is equal to $\mathbb{R}^n$ with probability 1. The paper also proposes an algorithm to draw such set D_k (p.14 and 15). To summarize, these two steps evolving at different scales allow a "first global pass" before doing a "second finer pass" to gain efficiency.

The intuition behind the convergence of such method is pretty easy and lies on the existence of a descent direction in a positive basis D. Let's prove it in the case of a continuously differentiable function with $\nabla f(x) \neq 0$ for some $x \in \mathbb{R}^n$. Let $w = -\nabla f(x)$, we have $w^T w > 0$. As D span $\mathbb{R}^n$ positively, one has

$$w = \sum_{i=1}^{n_D} \lambda_i d_i \text{ with } \forall i, \lambda_i \geq 0 \Longrightarrow w^T w = \sum_{i=1}^{n_D} \lambda_i w^T d_i > 0$$

From which we conclude that at least one of the scalars $w^T d_i, \ldots, w^T d_{n_d}$ has to be strictly positive. As a descent direction d is any $d \in \mathbb{R}^n$ such that $-\nabla f(x)^T d > 0$, we can directly conclude that there must exist a descent direction in a positive basis. Hence, as λ_k and μ_k will strictly decrease if no better point is found, the existence of this descent direction tells us that we will fall into a local optima as λ_k and μ_k decrease. It can also be proved for non-continuously differentiable functions under certain assumptions and for that we let you refer to the proof in [2].

Regarding the adaptation of this algorithm in the case of constrained optimization where $X \neq \mathbb{R}^n$, a better point will not only be a point decreasing the function value, but also a point respecting the constraints. By doing this we ensure that the algorithm will converge to a local optima lying in the authorised part of the space X.

B. Covariance Matrix Adaptation Evolution Strategy

Evolution Strategies are stochastic derivative-free methods. As all Evolution Algorithm, they are based on biological evolutionary concepts such as natural selection, generating at each stage a new population (the offspring) and then selecting the best individuals from that population. Each generation of new individuals is generated by stochastic variation of the current parents.

The Covariance Matrix Adaptation (CMA) Evolution Strategy was introduced in its current form in 2004 in [4] to optimize non-convex, non-smooth objective functions or functions with features that make algorithms using gradients inoperable (plateaus, ridges, ill-condition, etc).

Sampling, Selection and Recombination

It uses randomized black box optimization: a population of λ independent points are sampled following a multivariate normal distribution parametrized by the current parent individual as mean and the current covariance matrix. At step $t+1$:

$$\forall\, i \in \{1, ..., \lambda\} \quad x_i^{(t+1)} \sim m^{(t)} + \sigma^{(t)} \mathcal{N}(0, C^{(t)})$$

Then all these points by evaluated and ranked using the objective function such that $f(x_{1:\lambda}^{(t+1)}) \leq \cdots \leq f(x_{\lambda:\lambda}^{(t+1)})$. The μ best points are selected and used to compute a weighted mean which becomes the new parent individual $m^{(t+1)}$:

$$m^{(t+1)} = \sum_{i=1}^{\mu} w_i x_{i:\lambda}^{(t+1)} \quad \text{where} \quad \sum_{i=1}^{\mu} w_i = 1, \quad w_1 > \cdots > w_\mu > 0$$

Covariance Matrix Adaptation

The covariance matrix C and step-size control σ are then updated so that the distribution will converge to the global minimum. If it does not find the minimum, the covariance matrix will grow in the desired direction(s) (greatest eigenvalues in these directions and greatest step-size), otherwise it will shrink in the desired direction(s) until it is close enough to the minimum to trigger a stopping criteria. An illustration of this method optimizing a simple convex function over 10 iterations in shown in Figure 2.

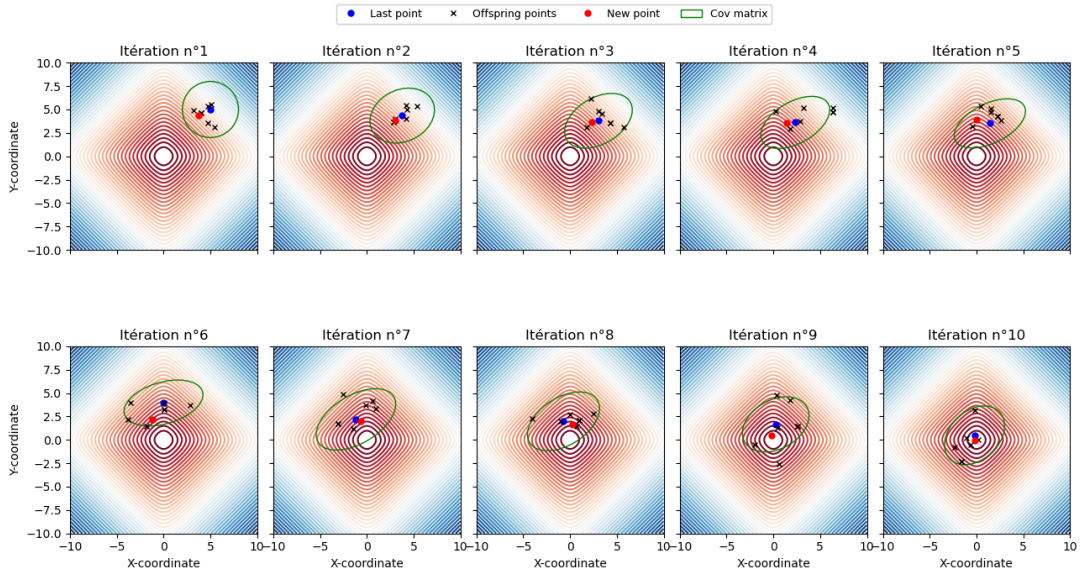

Fig. 2. Illustration of CMA-ES search mechanism with opportunistic approach for $f : (x, y) \to \sqrt{1+x^2} + \sqrt{1+y^2}$ in $\mathbb{R}^2$. Red represents lower values of f. The green error ellipse is an iso-contour of the Normal Distribution drawn using the covariance matrix eigenvectors and eigenvalues: it is one representation of the covariance matrix.

As stated by N. Hansen in [6], *"learning the covariance matrix in the CMA-ES is analogous to learning the inverse Hessian matrix in a quasi-Newton method"*. The covariance matrix adaptation uses two different updates: rank-1 update and rank-μ update.

$$C^{(t+1)} = (1 - c_1 - c_\mu)C^{(t)} + c_1 p_c^{(t+1)} p_c^{(t+1)T} + c_\mu R_\mu^{(t+1)}$$

The rank-1 update uses an evolution path p_c parametrized by c_c to store information about correlations between previous and current generations, and is especially usefull for small population (λ small). The rank-μ update takes information from the μ best points selected in the current generation and is larger and especially usefull for large population sizes.

The step-size σ is used to control the overall scale of the distribution: whereas the covariance matrix update may increase in only one or a few directions, the step-size controls the distribution scale in all directions. It prevents a too quick convergence by trying to keep the distribution scale at a reasonable level. Its update uses an evolution path parametrized by c_σ and a change parameter d_σ.

Change parameters

With CMA-ES, we have 4 change parameters or rates that we have to look to and adjust carefully [6]:

- The change parameter for the mean value $m^{(t)}$, determined by the parent number μ and the recombination weights w_i: $\mu_{eff} = \frac{1}{\sum_{i=1}^{\mu} w_i^2}$. If μ_{eff} is large, it means the possible change in m is small.
- The change parameters for covariance matrix: c_1 for rank-1-update and c_μ for rank-μ update, independents from population size λ and parent number μ.
- The change parameter for step-size control d_σ independent fom changes in C.

Although there are no guarantees on the convergence of CMA-ES, it has proven experimentally that it outperforms most current algorithms using gradients for reasonable dimensions (of the order of tens). Nevertheless, its C, sigma and m parameters are unbiased: under random selection, we always have $\mathbb{E}\left(\frac{1}{\lambda}\sum_{i=1}^{\lambda} x_i^{(t+1)} \middle| m^{(t)}\right) = m^{(t)}$. It is a guarantee of stationarity, since a bias would favour divergence or premature convergence of the algorithm [6].

Case of a constrained problem: augmented Lagrangian

In [1], A. Atamna, A. Auger and N. Hansen developped a method to handle a single linear constraint with CMA-ES using an augmented Lagrangian with penalty function:

$$\mathcal{L}(x, \gamma, \omega) = f(x) + \begin{cases} \gamma g(x) + \frac{\omega}{2} g^2(x) & \text{if } \gamma + \omega g(x) \geq 0 \\ -\frac{\gamma^2}{2\omega} & \text{otherwise} \end{cases}$$

Where ω is the penalty factor and γ the Lagrange factor.

If f and g are differentiable, then for x^* the optimum of the constrained problem and γ^* the optimal Lagrange factor, we have for all $\omega > 0$ [1]:

$$\nabla_x \mathcal{L}(x^*, \gamma^*, \omega) = \nabla_x f(x^*) + \max(0, \gamma^* + \omega g(x^*)) \nabla_x g(x^*) = 0$$

Hence our update for γ will be $\gamma^{(t+1)} = \max(0, \gamma^{(t)} + \omega(t) g(x(t+1)))$.

III. Implementation

A. Mesh Adaptive Direct Search

We used the implementation of LTMADS by C. Audet and J.E. Denis from [2]. Stopping criterias are detailed in appendix in Table I. First let's recall the general algorithm.

LTMADS is more specific in the way that it tells us how to choose μ_0 and λ_0, how to randomly draw D_k by ensuring its properties and how to process the **UPDATE** step. We choose $\mu_0 = \lambda_0 = 1$ and we will update μ_k in the following way:

$$\mu_{k+1} = \begin{cases} \frac{\mu_k}{4} & \text{if no better point if found during POLL step} \\ 4\mu_k & \text{if a better point is found and if } \mu_k \leq \frac{1}{4} \\ \mu_k & \text{otherwise} \end{cases}$$

Algorithm 1: General MADS algorithm with POLL step

Initialization: Choose $x_0 \in X$, $\lambda_0 \geq \mu_0$, D, $k = 0$, max_iter, f_{tol}, x_{tol}, λ_{min}, μ_{min} and ϵ

while *No Stopping Criteria is Satisfied* **do**

SEARCH: Try to find a better point by evaluating either all points in M_k, a deterministic subset of M_k, or a randomly drawn subset of M_k in an opportunistic approach or not.

POLL: Draw D_k and try to find a better point in P_k in an opportunistic approach or not.

UPDATE: If a better point has been found, increase μ_k and λ_k such that $\lambda_{k+1} \geq \mu_{k+1}$, $\lambda_{k+1} \geq \lambda_k$ and $\mu_{k+1} \geq \mu_k$. Otherwise, decrease them such that $\lambda_{k+1} \geq \mu_{k+1}$, $\lambda_k \geq \lambda_{k+1}$ and $\mu_k \geq \mu_{k+1}$.

end

Here, we ensure that μ_k will always be a power of 4 and never exceeds 1, and that $\frac{1}{\sqrt{\mu_k}} \geq 0$ will always be a non-negative power of 2 and hence an integer. For D_k to respect its properties, one direction need to be drawn differently from the others. For a given integer l, we call this direction $b(l) \in \mathbb{R}^n$ which is constructed as follows:

1. Draw $\hat{i}$ uniformly in $N = \{1, \ldots, n\}$.
2. Draw $b_{\hat{i}}(l)$ uniformly in $\{-2^l, 2^l\}$ and $\forall i \in N \setminus \{\hat{i}\}$, draw $b_i(l)$ uniformly in $\{-2^l+1, \ldots, 2^l-1\}$.

All $b(l)$ are stored somewhere such that once $b(l)$ is processed for some l, we use it again once we fall on the same l. Finally, D_k will be constructed as follows:

1. Let $l = -\log_4(\mu_k)$, construct or retrieve $b(l)$ and let $\hat{i}$ such that $b_{\hat{i}}(l) = \pm 2^l$.
2. Draw $L \in \mathbb{R}^{(n-1)\times(n-1)}$ a lower triangular matrix such that the elements on the diagonal are uniformly drawn in $\{-2^l, 2^l\}$ and the lower components are uniformly drawn in $\{-2^l+1, \ldots, 2^l-1\}$. L is a basis in $\mathbb{R}^{(n-1)}$.
3. With $\{p_1, \ldots, p_{n-1}\}$ a random permutation of $N \setminus \{\hat{i}\}$, set

$$B_{p_i,j} = L_{i,j} \text{ for } i, j = 1, \ldots, n-1$$
$$B_{\hat{i},j} = 0 \text{ for } j = 1, \ldots, n-1$$
$$B_{i,n} = b(l)_i \text{ for } i = 1, \ldots, n-1$$

   B is a basis in $\mathbb{R}^n$
4. Set B' the matrix resulting from the random permutation of the columns in B.
5. Either complete B' to a minimal positive basis $D_k = [B'\ d]$ with $d = -\sum_{j \in N} B'_{i,j}$ in which case we will update λ_k with $\lambda_k = n\sqrt{\mu_k} \geq \mu_k$, or complete B' to a maximal positive basis $D_k = [B'\ -B']$ in which case we will update λ_k with $\lambda_k = \sqrt{\mu_k} \geq \mu_k$

We therefore have everything to implement the LTMADS algorithm and apply it to some optimization problems.

B. Covariance Matrix Adaptation Evolution Strategy

We used the implementation by N. Hansen from [5] as a basis for our code. We extended it by using the concepts of Lagrangian for single linear constraint handling from [1] and applied this extension to some quadratic constraints.

The algorithm is made a repetition of steps until some criteria are met: see algorithm 2. Stopping criteria are detailed in appendix in Table I.

Algorithm 2: CMA-ES algorithm with or without Lagrangian constraint handling

Initialize constant parameters λ, μ, c_1, c_c, c_μ, c_σ, weights for recombination $(w_i)_{1\leq i\leq \mu}$, f_{tol}, x_{tol}, max_iter, set $t = 0$.

Initialize dynamic parameters evolution path $p_c = 0$, $p_\sigma = 0$ covariance matrix $C^{(0)} = \mathbf{I}$. Choose $x_0 \in X$. Choose step-size $\sigma^{(0)}$, Lagrange factor $\gamma^{(0)} \in \mathbb{R}$, penalty factor for Lagrangian $\omega^{(0)} \in \mathbb{R}_+^*$ adapted to the problem.

while *No Stopping Criteria is Satisfied* **do**

SAMPLE: Sample new population of search points following the normal multivariate distribution $m^{(t)} + \sigma^{(t)}\mathcal{N}(0, C^{(t)})$.

SELECTION AND RECOMBINATION Rank points according to objective function value (Lagrangian value if the problem is constrained). Select the best μ individuals. Compute the weighted mean to have $m^{(t+1)}$.

STEP-SIZE CONTROL Update the cumulation path p_σ and update the step-size $\sigma^{(t)}$.

COV MATRIX ADAPTATION Update the evolution path p_c and update the covariance matrix: $C^{(t+1)} = (1 - c_1 - c_\mu)C^{(t)} + c_1 p_c p_c^T + c_\mu R_\mu^{(t+1)}$.

LAGRANGIAN UPDATE If there is a constraint $g(x) \leq 0$: $\gamma^{(t+1)} = \gamma^{(t)} + \omega^{(t)} g(m^{(t+1)})$. If the constrained is not met ω is increased: the constraint takes more value in the Lagrangian. Otherwise, decrease ω.

end

The detailed steps and equations for CMA-ES are as follows:

1. Sampling: Sample λ individuals $x_k^{(t+1)} \sim m^{(t)} + \sigma^{(t)}\mathcal{N}(0, C^{(t)})$.

2. Selection and Recombination. Evaluate and rank the individuals by f evaluation: $f(x_{1:\lambda}^{(t+1)}) \leq \cdots \leq f(x_{\lambda:\lambda}^{(t+1)})$. If there is a constraint $g(x) \leq 0$, we rank by evaluation of the lagrangian $\mathcal{L}$: $\mathcal{L}(x_{1:\lambda}^{(t+1)}, \gamma^{(t)}, \omega^{(t)}) \leq \cdots \leq \mathcal{L}(x_{\lambda:\lambda}^{(t+1)}, \gamma^{(t)}, \omega^{(t)})$. Keep the μ best individuals and compute the *weighted average of μ selected points*:

$$m^{(t+1)} = \sum_{i=1}^{\mu} w_i x_{i:\lambda}^{(t+1)} \quad \text{where} \quad \sum_{i=1}^{\mu} w_i = 1, \quad w_1 > \cdots > w_\mu > 0 \tag{1}$$

3. Covariance Matrix Adaptation: update the evolution path and the covariance matrix. Here is the rank-1 update alone:

$$p_c^{(t+1)} = (1 - c_c)p_c^{(t)} + \sqrt{c_c(2 - c_c)\mu_{eff}}\frac{m^{(t+1)} - m^{(t)}}{\sigma^{(t)}} \tag{2}$$

$$C^{(t+1)} = (1 - c_1)C^{(t)} + c_1 p_c^{(t+1)} p_c^{(t+1)T} \tag{3}$$

Add the rank-μ update alone:

$$y_{i:\lambda}^{(t+1)} = \frac{x_{i:\lambda}^{(t+1)} - m^{(t)}}{\sigma^{(t)}}$$

$$C^{(t+1)} = (1 - c_\mu)C^{(t)} + c_\mu \sum_{i=1}^{\mu} w_i y_{i:\lambda}^{(t+1)} y_{i:\lambda}^{(t+1)T} \tag{4}$$

Thus combining rank-1 and rank-μ, the actual update is:

$$C^{(t+1)} = (1 - c_1 - c_\mu)C^{(t)} + c_1 \underbrace{p_c^{(t+1)} p_c^{(t+1)T}}_{\text{rank-1 update}} + c_\mu \underbrace{\sum_{i=1}^{\mu} w_i y_{i:\lambda}^{(t+1)} y_{i:\lambda}^{(t+1)T}}_{\text{rank-}\mu\text{ update}} \tag{5}$$

4. Update the step-size control parameter σ. As for the Covariance Matrix rank-one update, we will use an evolution path to keep in memory the past experiences:

$$p_\sigma^{(t+1)} = (1 - c_\sigma)p_\sigma^{(t)} + \sqrt{c_\sigma(2 - c_\sigma)\mu_{eff}}\, C^{(t)^{-\frac{1}{2}}} \frac{m^{(t+1)} - m^{(t)}}{\sigma^{(t)}} \tag{6}$$

Then the step-size control is updated using exponential:

$$\sigma^{(t+1)} = \sigma^{(t)} \exp\left(\frac{c_\sigma}{d_\sigma}\left(\frac{||p_\sigma^{(t+1)}||}{||\mathbb{E}\left[\mathcal{N}(\mathbf{0}, \mathbf{I})\right]||} - 1\right)\right) \tag{7}$$

5. Update Lagrangian parameters if the problem is constrained:

$$\gamma^{(t+1)} = \gamma^{(t)} + \omega^{(t)} g(m^{(t+1)})$$

$$\omega^{(t+1)} = \begin{cases} 2^{\frac{1}{4n}}\omega^{(t)} & \text{if } \omega^{(t)} g^2(m^{(t+1)}) < 3\frac{|\mathcal{L}(m^{(t+1)}, \gamma^{(t)}, \omega^{(t)}) - \mathcal{L}(m^{(t)}, \gamma^{(t)}, \omega^{(t)})|}{n} \\ & \text{or } 5|g(m^{(t+1)} - g(m^{(t)}| < |g(m^{(t)}| \\ 2^{-\frac{1}{n}}\omega^{(t)} & \text{otherwise.} \end{cases}$$

IV. Experiments

The code for all our experiments, functions, and optimizers can be found on Github.

The purposes of our experiments are to:

1. Show that CMA-ES and MADS offer an improvement in terms of convergence compared to classic methods (Gradient Descent, Newton).

2. Compare their performances on constrained problems. However, there are no good ways of handling constraints with CMA-ES, and the method we chose using augmented Lagrangian can only handle a single linear constraint following [1], but we tried it with a single quadratic constraint. Therefore our experiments will only deal with problems with a single linear or quadratic constraint. The constraints we used are in appendix in Table II.

3. Verify whether the curse of dimension is as much an issue in constrained problems as in unconstrained problems.

We used several classic test functions in optimization to verify the convergence of our algorithms on different characteristics. The functions draxn in dimension 2 for unconstrained problems are in fig:unconsfunc2d, and the ones for the constrained problem are in fig:consfunc. In dimension n, we used only Sphere, Rosenbrock, Rastigrin, and Styblinski-Tang for the unconstrained problem.

Here are the tricks in the functions we chose:

- Sphere, Norm1Sphere: baseline
- Rosenbrock: non-convex, thin parabolic valley
- Rastigrin: lots of local minima
- Styblinski-Tang: four very similar and close local minima, one global minimum
- Levy n.13: succession of ridges

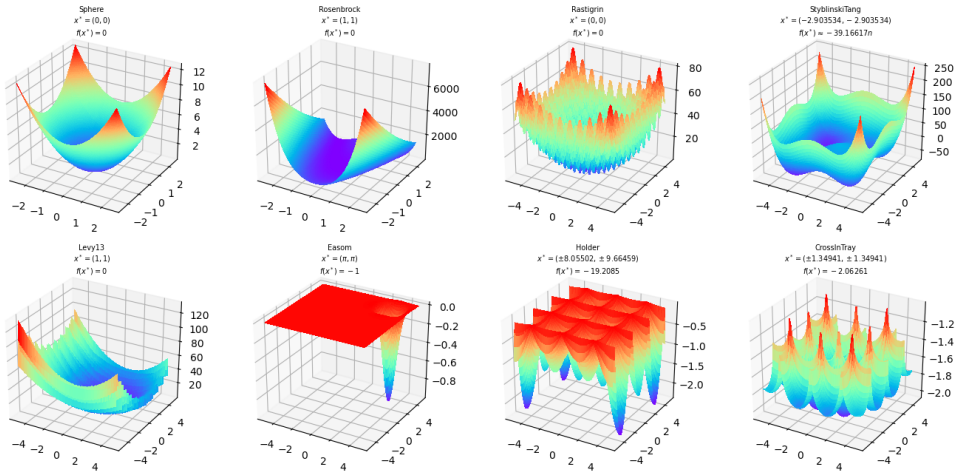

Fig. 3. Test functions for the unconstrained problem in dimension 2. Note that *Cross In Tray* and *Holder* do not have Gradients, hence we could not use with Gradient-Based methods.

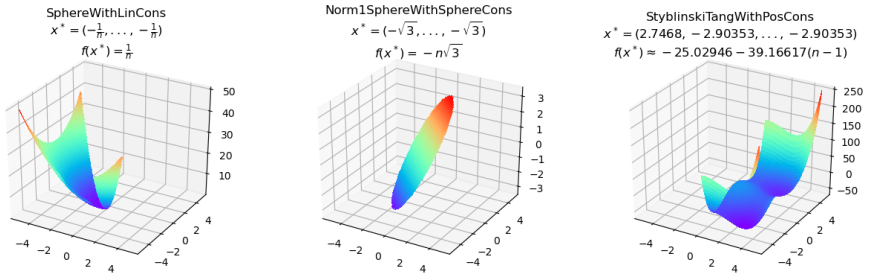

Fig. 4. Test functions for the constrained problem in dimension 2 and in dimension n.

- Easom: plateau with very thin hole where the global minimum
- Holder: waffle structure with lots of local minima
- Cross In Tray: high ridges and lots of local minima

The protocol is the following:

- Draw uniformly a great number of initial point x0 in the hypercube [-L, L]n (we chose L=50 for the final presentation, but here we will take L=20). For constrained functions consider x0 only if it lies in the definition domain.
- For each of these initial points, run each optimization method (or optimizer) on each function. See hyperparameters in appendix.
- During each optimization, track the elapsed time, $||x_{final} - x^*||$ and $|f(x_{final}) - f(x^*)|$, and the final number of iterations.
- Process means and confidence intervals for each pair of optimizer and function.

Note that for Gradient-Based optimization methods we used Newton's basic method and Gradient Descent for unconstrained problems, and Newton's method with line search and Newton's method with log-barrier function for constrained optimization.

V. Results and Conclusion

First for the unconstrained optimization problem in dimension 2 (Figure 5), CMA-ES outperforms all the other algorithms on all functions except MADS on Rastigrin function, and Holder function for which no optimizer had significant results (Table III). On the baseline function Sphere, CMA-ES run in the same range time as gradient descent. CMA-ES is slower than MADS by a factor 2 to 5 on Sphere, Rastigrin, Styblinski-Tang, Levy13 and Easom (tab:time$_u ncons_2$).

For the constrained problem in dimension 2 (Figure 7), CMAES performs well on simple functions but suprisingly very badly on Syblinski-Tang, and is still very slow (Table IX, Table X). On the contrary, MADS achieve poor convergence performances on simple function such as Sphere and Norm1Spehre, but converges 45% of the time on Syblinski-Tang. CMAES achieves to converge statistically as often as he newton method with a log barrier, but in a much slower way. MADS and Newton's method with line search did not perform well because they tend to violate the constraints very easily, triggering one of the stopping criterias.

For unconstrained optimization problem in dimension 10 (Figure 6), CMAES, and MADS to a lesser extent, showed a huge drop in convergence performances on Rosenbrock and Rastigrin. However their performances remained much better than gradient descent and basic Newton's method (Table VI). On top of that time ellapsed during each optimization was on average 5 times times ellapsed per optimization in dimension 2 for CMAES (Table VII): the curse of dimension starts to is starting to emerge, although not yet in exponential time: for λ points sampled in dimension 1, you need λ^n points in dimension n!

For the constrained problem in dimension n, we observe a combinations of observations (Figure 8): CMAES and Newton's method with log-barrier achieve again statistically similar convergence performances, except for Syblisnki-Tang for which as always, CMAES is bad whereas Newton's method converges 8% of the time (Table XII). MADS performs very poorly (not a single optimization converged). CMAES against suffer from very slow optimizations on average (Table XIII).

To conclude, although Derivative-free optimization methods such as MADS and CMAES achieve much greater performances than gradient based methods, they tend to be slower. These two factors, higher converging capability and higher optimization duration, are exacerbated for CMAES in unconstrained optimization. However those algorithms are not efficiently designed to handle constraints, either single or multiple ones, even if some research is ongoing. Finally, these algorithms suffer of the curse of dimension whether in theory or in practice and will need to correct this huge drawback to become usefull in real world problems such as Operation Research or Machine Learning, where the number of parameters is on a completely different scale.

References

[1] Asma Atamna, Anne Auger, and Nikolaus Hansen. "Augmented Lagrangian Constraint Handling for CMA-ES—Case of a Single Linear Constraint". In: *Proceedings of the 14th International Conference on Parallel Problem Solving from Nature*. Edinburgh, United Kingdom, Sept. 2016, pp. 181 –191. DOI: 10.1007/978-3-319-45823-6_17. URL: https://hal.inria.fr/hal-01390386.

[2] Charles Audet and John E Dennis Jr. "Mesh adaptive direct search algorithms for constrained optimization". In: *SIAM Journal on optimization* 17.1 (2006), pp. 188–217.

[3] Stephen Boyd, Lin Xiao, and Almir Mutapcic. "Subgradient methods". In: *lecture notes of EE392o, Stanford University, Autumn Quarter* 2004 (2003), pp. 2004–2005.

[4] N. Hansen and S. Kern. "Evaluating the CMA Evolution Strategy on Multimodal Test Functions". In: *Parallel Problem Solving from Nature PPSN VIII*. Ed. by X. Yao et al. Vol. 3242. LNCS. Springer, 2004, pp. 282–291.

[5] Nikolaus Hansen. *Pure CMA-ES*. code. [Online; accessed 13-April-2020]. 2003. URL: http://cma.gforge.inria.fr/purecmaes.m.

[6] Nikolaus Hansen. "The CMA Evolution Strategy: A Tutorial". In: *CoRR* abs/1604.00772 (2016). arXiv: 1604.00772. URL: http://arxiv.org/abs/1604.00772.

APPENDIX

A. Stopping Criteria, Convergence Criteria, Constraints, Hyperparameters

Stopping criteria

We had to add a lot of stopping criterias for all functions. Below is a comprehensive list of these and how they operate:

- Maximum number of iteration max_iter.
- Tolerance f_{tol} such that we stop the algorithm as soon as $|f(x_{t+1}) - f(x_t)| < f_{tol}$.
- Tolerance x_{tol} such that we stop the algorithm as soon as $\|x_{t+1} - x_t\| < x_{tol}$.
- Tolerance g_{tol} such that we stop the algorithm as soon as $\|g(x_t)\| < g_{tol}$.
- Tolerance div_{tol} such that we stop the algorithm as soon as $\|x_{t+1}\| > div_{tol}$ or $|f(x_{t+1})| > div_{tol}$.
- Tolerance ϵ. For Newton's method with line-search, it is the tolerance on the value of the optimal α in the bisection algorithm: if $|\alpha_u - \alpha_l| < \epsilon$ then we end the bisection algorithm.
- If the constraint is violated, the algorithm stops. For Newton's method with Log-barrier function, there is a tolerance ϵ on the compliance with the constraint. Since for CMA-ES the augmented Lagrangian handles constraints dynamically and with a penalty factor, if a constraint is not met the mean will come back into the feasible zone on its own and there is not point in stopping the algorithm for unmet constraints.
- For MADS only, thresholds λ_{min}, μ_{min} or ϵ such that we stop the algorithm as soon as either $\lambda_{k+1} < \lambda_{min}$, $\mu_{k+1} < \mu_{min}$, $\lambda_{k+1} < \epsilon$ or $\mu_{k+1} < \epsilon$.

The stopping criteria parameters were almost always as follows: $f_{tol} = 1e-14$, $x_{tol} = 1e-14$, $g_{tol} = 1e-14$, $div_{tol} = 1e10$, $\epsilon = 1e-8$, $max\_iter = 1000$ for unconstrained optimizations and $max\_iter = 2000$ for constrained ones. In the following table, we indicate the optimizer and the functions for which these parameters changed.

TABLE I
STOPPING CRITERIAS CHANGES COMPARED TO USUAL FOR EACH OPTIMIZER AND FOR EACH FUNCTION CONCERNED

Function	Stopping criteria changes	Optimizer	Stopping criteria changes
StyblinskiTang	$x_{tol} = 0$	MADS	$\mu_{min,uncons} = (4^{14})^{-1}$, $\mu_{min,cons} = (4^{12})^{-1}$
Easom	$f_{tol} = 0$, $max\_iter = 2000$		$\lambda_{min} = 0$, $\epsilon = 0$
Holder	$f_{tol} = 0$	CMA-ES	no constraints violation stopping criteria
StyblinskiTangWithPosCons	$x_{tol} = 0$		

Convergence criteria

For all experiments and optimizations we considered that an optimization was successful (it converged) when the two following criteria were met:

1. $|x_{final} - x^*| < 1e-3$.
2. $|f(x_{final}) - f(x^*)| < 1e-3$.

Please note that we computed the modified minima and objective function optimal value for constrained problems.

Constraints

TABLE II
CONSTRAINTS USED FOR THE CONSTRAINED PROBLEM

Function	Constraint
SphereWithLinCons	$\sum_{i=1}^{n} x_i \leq -1$
Norm1SphereWithSphereCons	$\sum_{i=1}^{n} x_i^2 \leq 3n$
StyblinskiTangWithPosCons	$-x_0 \leq 0$

Hyperparameters

A lot of Hyperparameters are needed for CMA-ES and we almost always used those provided by N. Hansen in [6] p27. The only exceptions are for functions Rastigrin, Styblinski-Tang, and Easom, for which we used $\lambda = \lfloor 2 \log n \times (4 + \lfloor 3 \log n \rfloor) \rfloor$ instead of $\lambda = 4 + \lfloor 3 \log n \rfloor$ where n is the dimension.

For MADS, we took initial parameters $\mu_0 = 1$ and $\lambda_0 = 1$ as in the paper by C. Audet [2].

Confidence Intervals

All confidence intervals in the results tables are 95% confidence intervals computed as follows, for each series of result for one pair (optimizer, function):

$$x \pm Z_{\alpha/2} \frac{\sigma}{\sqrt{n}} \text{ where } \begin{cases} \sigma \text{ is the standard deviation} \\ n \text{ is the number of data points} \\ Z_{0.95/2} \approx 1.92 \end{cases}$$

B. Unconstrained Problem, n=2

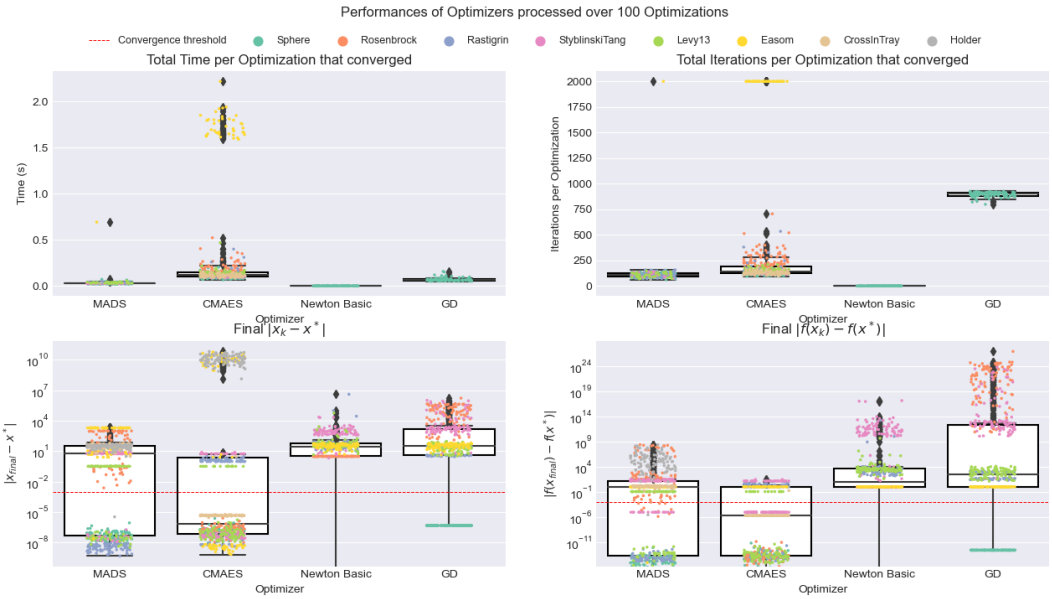

Fig. 5. Convergence over 100 optimizations (for clarity) for each optimizer and function for the unconstrained problem in dimension 2. The graphs on the second row are showing the convergence in terms of distance to x_{opt} (left) and distance to f_{opt}: an optimization is considered as successful if at the end these two distance are less than $1e-3$. The graphs on the first row are showing, for each successfull optimization, the duration and number of optimization steps needed to reach the minimum. We capped the number of iterations at 1000, as there is not much evolution after.

TABLE III
PERCENTAGE OF OPTIMIZATION THAT CONVERGED, UNCONSTRAINED, N=2, OVER 1000 OPTIMIZATIONS

Optimizer	Sphere	Rosenbrock	Rastigrin	StyblinskiTang	Levy13	Easom	CrossInTray	Holder
MADS	100.0%±0.0%	0%±0%	100.0%±0.0%	26.4%±3.6%	40.7%±3.4%	7.5%±2.3%	8.2%±2.9%	7.5%±2.3%
CMAES	100.0%±0.0%	100.0%±0.0%	20.4%±2.3%	65.4%±3.4%	87.8%±2.2%	79.5%±2.1%	99.0%±1.0%	0%
Newton Basic	100.0%±0.0%	0%	0%	0%	0%	0%±0%	0%	0%
GD	100.0%±0.0%	0%	0%	0%	0%	2.2%±1.1%	0%	0%

TABLE IV
AVERAGE DURATION OF OPTIMIZATIONS THAT CONVERGED (TIME IN SECONDS), UNCONSTRAINED, N=2, OVER 1000 OPTIMIZATIONS

Optimizer	Sphere	Rosenbrock	Rastigrin	StyblinskiTang	Levy13	Easom	CrossInTray	Holder
MADS	0.023s±0.000s	0.305s±0.002s	0.034s±0.001s	0.029s±0.001s	0.031s±0.001s	0.674s±0.006s	0.025s±0.001s	0.261s±0.011s
CMAES	0.089s±0.001s	0.165s±0.003s	0.204s±0.005s	0.122s±0.002s	0.138s±0.002s	1.690s±0.019s	0.098s±0.001s	/
Newton Basic	0.000s±0.000s	/	/	/	/	0.002s±0.000s	/	/
GD	0.062s±0.001s	/	/	/	/	0.114s±0.002s	/	/

TABLE V
AVERAGE NUMBER OF ITERATIONS PER OPTIMIZATION THAT CONVERGED, UNCONSTRAINED, N=2, OVER 1000 OPTIMIZATIONS

Optimizer	Sphere	Rosenbrock	Rastigrin	StyblinskiTang	Levy13	Easom	CrossInTray	Holder
MADS	66±0	1000±0	76±1	67±0	68±0	2000±0	56±0	676±27
CMAES	111±1	206±3	210±4	130±1	145±1	2000±0	120±1	/
Newton Basic	1±0	/	/	/	/	5±0	/	/
GD	815±2	/	/	/	/	1050±12	/	/

C. Unconstrained Problem, n=10

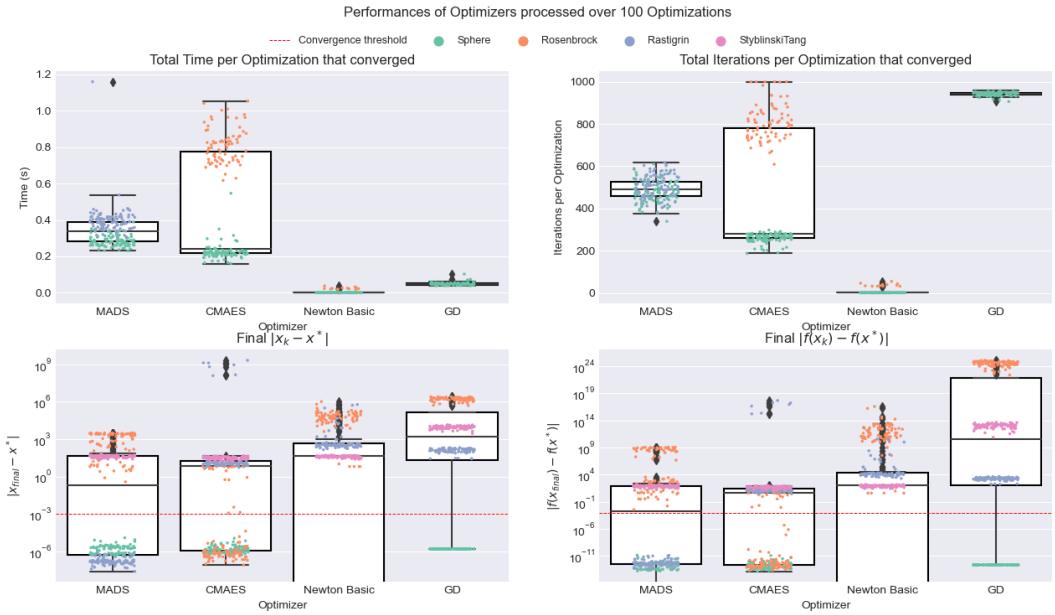

Fig. 6. Convergence over 100 optimizations (for clarity) for each optimizer and function for the unconstrained problem in dimension 10. The graphs on the second row are showing the convergence in terms of distance to x_{opt} (left) and distance to f_{opt}: an optimization is considered as successful if at the end these two distance are less than $1e-3$. The graphs on the first row are showing, for each successfull optimization, the duration and number of optimization steps needed to reach the minimum. We capped the number of iterations at 1000, as there is not much evolution after.

TABLE VI
PERCENTAGE OF OPTIMIZATION THAT CONVERGED, UNCONSTRAINED, N=10, OVER 1000 OPTIMIZATIONS

Optimizer	Sphere	Rosenbrock	Rastigrin	StyblinskiTang
MADS	100.0%±0.0%	0%	100.0%±0.0%	0%
CMAES	100.0%±0.0%	89.6%±1.9%	0%	2.1%±0.9%
Newton Basic	100.0%±0.0%	50.9%±3.0%	0%	0%
GD	100.0%±0.0%	0%	0%	0.3%±0.3%

TABLE VII
AVERAGE DURATION OF OPTIMIZATIONS THAT CONVERGED (TIME IN SECONDS), UNCONSTRAINED, N=10, OVER 1000 OPTIMIZATIONS

Optimizer	Sphere	Rosenbrock	Rastigrin	StyblinskiTang
MADS	0.273s±0.004s	/	0.389s±0.005s	/
CMAES	0.249s±0.003s	1.036s±0.014s	/	0.284s±0.002s
Newton Basic	0.000s±0.000s	0.033s±0.001s	/	/
GD	0.054s±0.001s	/	/	0.005s±0.000s

TABLE VIII
AVERAGE NUMBER OF ITERATIONS PER OPTIMIZATION THAT CONVERGED, UNCONSTRAINED, N=10, OVER 1000 OPTIMIZATIONS

Optimizer	Sphere	Rosenbrock	Rastigrin	StyblinskiTang
MADS	273±1	/	307±1	/
CMAES	241±1	768±5	/	275±1
Newton Basic	1±0	35±1	/	/
GD	862±0	/	/	58±0

D. Constrained Problem, n=2

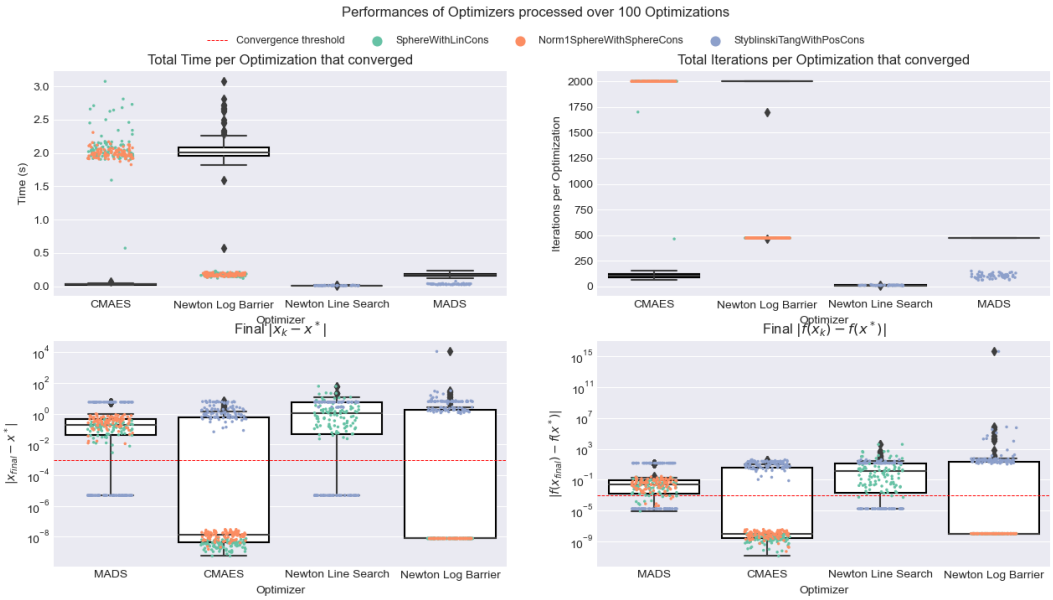

Fig. 7. Convergence over 100 optimizations (for clarity) for each optimizer and function for the constrained problem in dimension 2. The graphs on the second row are showing the convergence in terms of distance to x_{opt} (left) and distance to f_{opt}: an optimization is considered as successful if at the end these two distance are less than $1e-3$. The graphs on the first row are showing, for each successfull optimization, the duration and number of optimization steps needed to reach the minimum. We capped the number of iterations at 1000, as there is not much evolution after.

TABLE IX
PERCENTAGE OF OPTIMIZATION THAT CONVERGED, CONSTRAINED, N=2, OVER 1000 OPTIMIZATIONS

Optimizer	SphereWithLinCons	Norm1SphereWithSphereCons	StyblinskiTangWithPosCons
MADS	0.4%±0.4%	0.2%±0.3%	44.5%±3.0%
CMAES	100.0%±0.0%	100.0%±0.0%	0%
Newton Line Search	0.3%±0.3%	0%	40.9%±3.0%
Newton Log Barrier	100.0%±0.0%	100.0%±0.0%	0.3%±0.3%

TABLE X
AVERAGE DURATION OF OPTIMIZATIONS THAT CONVERGED (TIME IN SECONDS), CONSTRAINED, N=2, OVER 1000 OPTIMIZATIONS

Optimizer	SphereWithLinCons	Norm1SphereWithSphereCons	StyblinskiTangWithPosCons
MADS	0.031s±0.001s	0.022s±0.000s	0.027s±0.000s
CMAES	1.037s±0.006s	0.978s±0.002s	/
Newton Line Search	0.024s±0.000s	/	0.011s±0.000s
Newton Log Barrier	0.195s±0.002s	0.211s±0.001s	0.127s±0.001s

TABLE XI
AVERAGE NUMBER OF ITERATIONS PER OPTIMIZATION THAT CONVERGED, CONSTRAINED, N=2, OVER 1000 OPTIMIZATIONS

Optimizer	SphereWithLinCons	Norm1SphereWithSphereCons	StyblinskiTangWithPosCons
MADS	82±1	60±1	68±0
CMAES	998±2	999±1	/
Newton Line Search	10±0	/	11±0
Newton Log Barrier	472±0	472±0	339±0

E. Constrained Problem, n=10

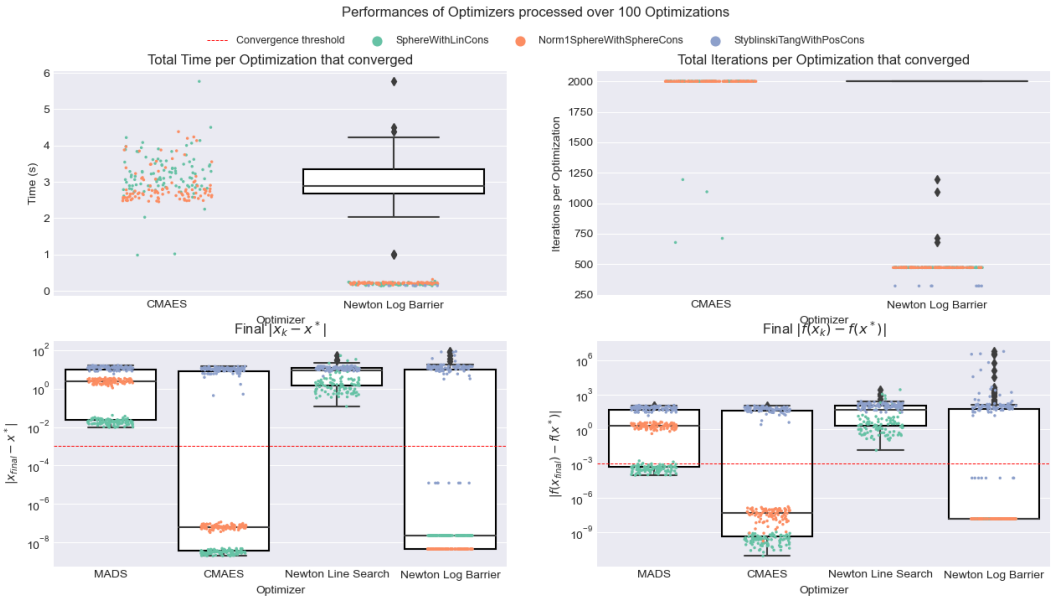

Fig. 8. Convergence over 100 optimizations (for clarity) for each optimizer and function for the constrained problem in dimension 2. The graphs on the second row are showing the convergence in terms of distance to x_{opt} (left) and distance to f_{opt}: an optimization is considered as successful if at the end these two distance are less than $1e-3$. The graphs on the first row are showing, for each successfull optimization, the duration and number of optimization steps needed to reach the minimum. We capped the number of iterations at 1000, as there is not much evolution after.

TABLE XII
PERCENTAGE OF OPTIMIZATION THAT CONVERGED, CONSTRAINED, N=10, OVER 100 OPTIMIZATIONS

Optimizer	SphereWithLinCons	Norm1SphereWithSphereCons	StyblinskiTangWithPosCons
MADS	0%	0%	0%
CMA-ES	100.0%±0.0%	100.0%±0.0%	0%
Newton Line Search	0%	0%	0%
Newton Log Barrier	100.0%±0.0%	100.0%±0.0%	8.0%±3.3%

TABLE XIII
AVERAGE DURATION OF OPTIMIZATIONS THAT CONVERGED (TIME IN SECONDS), CONSTRAINED, N=10, OVER 100 OPTIMIZATIONS

Optimizer	SphereWithLinCons	Norm1SphereWithSphereCons	StyblinskiTangWithPosCons
MADS	/	/	/
CMA-ES	3.201s±0.122s	2.890s±0.087s	/
Newton Line Search	/	/	/
Newton Log Barrier	0.191s±0.004s	0.212s±0.004s	0.144s±0.004s

TABLE XIV
AVERAGE NUMBER OF ITERATIONS PER OPTIMIZATION THAT CONVERGED, CONSTRAINED, N=10, OVER 100 OPTIMIZATIONS

Optimizer	SphereWithLinCons	Norm1SphereWithSphereCons	StyblinskiTangWithPosCons
MADS	/	/	/
CMA-ES	1957±29	2000±0	/
Newton Line Search	/	/	/
Newton Log Barrier	472±0	472±0	321±0