UCLouvain

epl

École polytechnique de Louvain

Numerical performance of non-monotone line searches on unconstrained global optimization problems

Author: **Romain Chauvaux**
Supervisor: **Geovani Nunes Grapiglia**
Readers: **Pierre-Antoine Absil, Laurent Jacques**
Academic year 2022–2023
Master [120] in Mathematical Engineering

Abstract

In this work, we investigate the use of non-monotone line search methods to approximately solve unconstrained global optimization problems. Firstly, we present an extensive numerical comparison between monotone and non-monotone line search methods. The results show that non-monotone line search methods outperform the monotone Armijo line search, providing significantly better approximate solutions, with iterates exhibiting the ability to escape from nearby non-global local minimizers. Secondly, we develop a new non-monotone line search for problems in which the optimal function value is known a priori. We find numerically that this new method outperforms existing monotone and non-monotone methods in instances of the Molecular Distance Geometry Problem.

Acknowledgements

First, I want to sincerely acknowledge my promoter Geovani Nunes Grapiglia for all his help during my master's thesis. He has always been available to discuss any step and directions we can explore. His commentary and advice were precious, and I am very grateful. I thank him again for all the time invested in meetings and discussions.

I also wish to thank the other students with whom I worked. Thank you for the interesting discussion and the pleasant work atmosphere.

I thank my family for all the help and support they give me during my studies. I specifically thank my girlfriend Anne-Lise for her unwavering presence. I wouldn't have been able to accomplish this work without you.

Finally, I would like to thank my mathematical teacher Mrs. Caroline Manfroid who gave me an interest in and passion for mathematics.

Contents

LIST OF FIGURES

Acronyms

BFGS Broyden-Fletcher-Goldfarb-Shanno method. 4, 10, 14, 24, 36, 37

DGP Distance Geometry Problem. 44

GW Griewank function. i, 25–27

M1 Monotone method 1. ii, 5, 25, 26, 31–34, 44, 48, 51, 52

MDGP Molecular Distance Geometry Problem. 4, 44–46

NM1 Non-monotone method 1. 19, 25, 26, 31, 32, 35, 37

NM2 Non-monotone method 2. 25, 26, 31, 32, 35, 37

NM3 Non-monotone method 3. i, ii, 4, 5, 20, 25–28, 30–34, 35, 37, 44, 48, 49, 51, 52

NMI Non-monotone informed method. ii, 5, 38, 39, 43, 44, 45, 48–53

NMR Nuclear magentic resonance spectroscopy. 44, 47, 50

SA Simulated Annealing. i, ii, 4, 5, 27, 33, 51

SR1 Symmetric rank-one method. 9, 10

SVD Singular value decomposition. 52

Introduction

Optimization problems occur in almost all areas of science and technology. For example, they appear in practical life examples such as airline operations or scheduling. We can also find them in industrial processes to reduce costs or increase efficiency. One of the famous optimization problems addressed in this paper is the structure of proteins.

Practical optimization methods are iterative schemes. For example, consider the problem of minimizing a function $f : \mathbb{R}^n \longrightarrow \mathbb{R}^n$. An optimization method starts with an initial guess x_0 and uses it to find a new iterate x_1. Doing this again will hopefully find an interesting solution x_n where n is sufficiently large.

Line search methods are one of the main classes of optimization methods. All line search methods respect the same framework to perform an iteration. It is defined for each iteration k by a direction d_k and a step size α_k.

$$x_{k+1} = x_k + \alpha_k d_k$$

Therefore, the iterate x_{k+1} is found thanks to the previous iterate x_k and a step α_k in a direction p_k. This line update justifies the name of the class. The choice of the direction d_k significantly influences the method's behavior, a first possibility is to consider the steepest descent $d_k = -\nabla f(x_k)$. The choice of step size α_k is also essential, especially for the convergence of the method. One possible choice is to consider an Armijo line search. The idea is to find for each iteration k a step size α_k that sufficiently decreases the objective function. More precisely, the Armijo line search tries to find a step size α_k that satisfies the rule (I.1).

$$f(x_k + \alpha_k d_k) \leq f(x_k) + c\alpha_k d_k^T \nabla f(x_k) \tag{I.1}$$

with $c \in (0, 1)$. If the rule (I.1) is not verified, we try again with $\alpha_k = \tau\alpha_k$ for a certain constant $\tau \in (0, 1)$.

The Armijo line search is monotone, in the sense that each iterate x_{k+1} gives a function value smaller than x_k, as we can see in the rule (I.1). In fact, the term $c\alpha_k d_k^T \nabla f(x_k)$ is always negative for $d_k = -\nabla f(x_k)$.

One of the main drawbacks of monotone line search is that its iterations tend to converge to local minimizers. This is a problem if we try to minimize a function with

many non-global local minimizers, such as the one shown in Figure I-1.

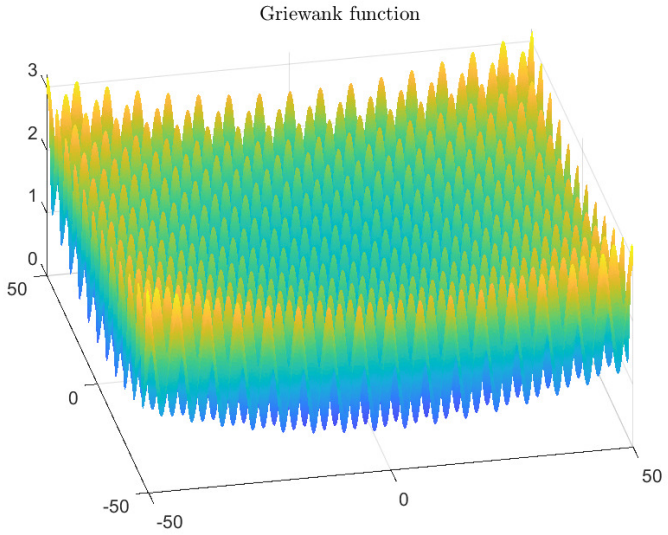


Figure I-1: The Griewank function

One way to deal with this type of problem is to use non-monotone line search methods, where the step size α_k is chosen so that

$$f(x_k + \alpha_k d_k) \leq f(x_k) + c\alpha_k d_k^T \nabla f(x_k) + v_k.$$

This type of line search makes it possible to accept x_{k+1} with $f(x_{k+1}) > f(x_k)$ thanks to the term v_k. The non-monotone behavior allows the iterates to "jump hills" and possibly escape from nearby non-global local minimizers.

Recently, Grapiglia and Sachs [1, 2] established convergence and complexity results for the non-monotone line search methods under the assumption that

$$\lim_{T \to +\infty} \frac{1}{T} \sum_{k=0}^{T-1} \nu_k = 0.$$

This condition is weak enough to allow the development of a wide range of non-monotone line search methods with convergence and worst-case complexity guarantees. For example, a new non-monotone line search inspired by the Simulated Annealing method has been proposed in [2]. When applied to the function in Figure 1, this new non-monotone line search method showed a remarkable superiority over state-of-the-art line search methods, providing significantly better approximate solutions.

In this work, we further investigate the efficiency of non-monotone line search. In particular, we present the results of extensive numerical experiments on a large set of benchmark global optimization problems. The experiments were conducted to understand when non-monotone line search methods show superior performance and to identify the best non-monotone strategies for problems with many non-global local

minima. Furthermore, exploiting the flexibility of the general scheme proposed in [2], we develop a new non-monotone line search method for problems where the optimal value is known. A relevant example is the Molecular Distance Geometry problem. This problem aims to find the relative positions of a set of atoms with the knowledge of some distances between them. The objective function is written as an error function with the distances of the atoms currently found and the exact value. The minimal value of the Molecular Distance Geometry Problem is 0 when the relative positions of the atoms perfectly match the set of distances available.

This work is organized as follows. Chapter 1 states the basics of optimization problems and introduces the methods used in the following chapters. First, we go through the classical optimization methods such as the Gradient, Newton, and Quasi-Newton methods and the monotone different line search methods. Then we visit the already existing non-monotone line search methods. Finally, we present the general framework proposed in [2] and further develop this new method based on Simulated Annealing.

In Chapter 2, we perform numerical investigations to compare the monotone and non-monotone line search algorithms. We first define clear test problems and optimize some parameters of the methods. Then, we highlight the behavior of each algorithm using data and performance profiles as well as box plots. Finally, we compare their performance with the well-known Simulated Annealing.

Chapter 3 defines the new non-monotone line search method capable of exploiting the optimal value of a problem. After a justification of its characteristics, a new algorithm is presented. A worst-case complexity analysis of this algorithm is carried out before numerical experiments with the previous methods.

Finally, Chapter 4 highlights the performance of our different methods on an actual application, the Molecular Distance Geometry Problem. First, some investigations into the structure of this problem are made. Then, numerical experiments are performed on benchmark problems of this type. Finally, we test our methods on real proteins from the Protein Data Bank [3] and analyze the quality of the solutions obtained.

Chapter 1

State of the Art

This chapter introduces the topic of global optimization problems. It gives an overview of some basic definitions and theorems. We go through some popular methods, namely the gradient method, Newton's method, and the BFGS method. A presentation of monotone line search methods is proposed, thanks to the Armijo line search and others. Then the history of non-monotone line search methods is presented with Grippo-Lampariello-Lucidi [4] and Zhang and Hager [5]. A general framework proposed by Grapiglia and Sachs [1, 2] is introduced, as well as a new non-monotone method inspired by Simulated Annealing. Furthermore, some convergence results are reviewed, and worst-case complexity is presented. The chapter ends with initial expectations about non-monotone line search methods compared to monotone ones.

1.1 Definitions and Basic Results

Let introduce $f : \mathbb{R}^n \to \mathbb{R}$ be a function that associates any points $x \in \mathbb{R}^n$ with the function value $f(x)$. From this function, we can define the two following types of minimizers,

Definition 1.1.1 (Local minimizer)**.** x^* is a local minimizer if $\exists B(x^*, \epsilon)$, a ball center at x^* with radius $\epsilon \geq 0$, such that $f(x) \geq f(x^*)\ \forall\, x \in B(x^*, \epsilon)$.

Definition 1.1.2 (Global minimizer)**.** x^* is a global minimizer if $f(x) \geq f(x^*)\ \forall\, x \in \mathbb{R}^n$

Finding a minimizer of a function is a challenge that occurs in many different situations. This search is represented mathematically by an optimization problem. In this Master's thesis, we will focus on the unconstrained optimization problems. The following formulation completely describes this class of problems.

$$\min_{x \in \mathbb{R}^n} f(x) \tag{1.1}$$

This problem is called unconstrained because it has no constraints on the variable x.

This optimization problem aims to find a global minimizer of the objective function f. However, ensuring its existence is only sometimes possible as f may not attain its minimum value, for example, $f(x) = x$. It is still possible to prove the existence of a global minimizer under some conditions. We will state these conditions below.

Theorem 1.1 (Weierstrass extreme value theorem)**.** *If f is a continuous function on a compact (closed and bounded) set, then it reaches its extreme values.*

Definition 1.1.3 (Coercive function)**.** f is said to be a coercive function if $\exists \nu \in :\{x \in P : f(x) \leq \nu\}$ is bounded with P the sublevel sets of f.

With these two elements, the following theorem 1.2 can be proved.

Theorem 1.2. *Let f be a continuous function on $\mathbb{R}^n$; if f is coercive, then f has at least one global minimizer.*

The theorem 1.2 is sufficient but not necessary. The coercivity of a function is not necessary to have a global minimizer, but it can easily be checked. Unfortunately, this proof of existence is not constructive, i.e. it doesn't give steps to find these global minimizers for coercive functions. The optimality conditions defined below have been defined to help us in this sense. Before using the derivative of f and all that follows, we must restrict the possible functions to smooth ones.

Definition 1.1.4 (Smooth functions)**.** A function is said to be smooth over a domain if it has continuous derivatives up to a desired order k on that domain. A standard notation is $f \in \mathcal{C}^k$ in the domain D.

The desired degree of smoothness depends on the problem and can be limited to one or two in our cases. Let's look at some essential examples of smooth and non-smooth functions. Here is a non-exhaustive list of standard smooth functions: polynomial, trigonometric, and norm l2 functions. Here is a non-exhaustive list of standard non-smooth functions: norm l^1, norm l^∞ and min-max functions.

We now state the first-order optimality condition theorem.

Theorem 1.3 (First-Order Optimality Condition)**.** *If $f \in \mathcal{C}^1$, then a necessary condition for the candidate x to be a local optimizer is,*

$$\nabla f(x) = 0$$

As the theorem states, this condition is a necessary condition for finding local minimizers. Every point that satisfies this condition is a critical point. Therefore, every local minimizer is a critical point, but the converse is untrue. Moreover, the first-order optimality condition does not give any specific information about global minimizers, except that they are also local minimizers by definition. Nevertheless, under certain conditions on the problem, we can use this theorem to have information about global minimizers.

We introduce here another concept, the convexity of a problem. The problem (1.1) is considered convex if the function f is convex.

Definition 1.1.5 (Convex functions)**.** A function $f : \mathbb{R}^n \longrightarrow \mathbb{R}$ is said to be convex if

the following property holds,

$$f(tx + (1-t)y) \leq tf(x) + (1-t)f(y)$$

where $x, y \in \mathbb{R}^n$ and $t \in (0,1)$.

If we work with smooth functions, we can establish the following lemma.

Lemma 1.4 (Differentiable convex function)**.** *Let f be differentiable, i.e. $f \in \mathcal{C}^1$, then f is convex if and only if*

$$f(x) \geq f(y) + \nabla f(y)^T (x - y)$$

Thanks to the concept of convexity, we can establish a very useful theorem for convex function.

Theorem 1.5. *Let f be a convex function; if x is a critical point, i.e. respects condition 1.3, then x is a global minimizer of f.*

Theorem 1.5 gives a necessary and sufficient condition for finding the global optimizer of a convex function f. This global optimizer is also the solution to the problem 1.1 associated with f since it is unconstrained.

In this Master's thesis, the focus is on non-convex functions where this last theorem does not hold. Indeed, if f is non-convex, the critical points can take various forms. We already know local and global minimizers, but there are also local maximizers, global maximizers and saddle points.

Definition 1.1.6 (local maximizer)**.** x^* is a local maximizer of 1.1 if $\exists B(x^*, \epsilon)$, a ball center at x^* with radius $\epsilon \geq 0$, such that $f(x) \leq f(x^*)\ \forall\, x \in B(x^*, \epsilon)$.

Definition 1.1.7 (global maximizer)**.** x^* is a global maximizer of 1.1 if $f(x) \leq f(x^*)\ \forall\, x \in \mathbb{R}^n$

Definition 1.1.8 (saddle point)**.** x^* is a saddle point if it respects condition 1.3 but is neither a local minimizer nor a local maximizer.

These possibilities complicate the search for global minimizers. Hopefully, some condition exists to help us find the nature of critical points. These are called second-order optimality conditions, as they use the second-order derivative of the function.

Theorem 1.6 (Second-Order Optimality Conditions)**.** *If $f \in \mathcal{C}^2$, then the following property holds for x^* such that $\nabla f(x^*) = 0$,*

- *If $\nabla^2 f(x^*) \succ 0$, then x^* is a local or global minimizer.*
- *If $\nabla^2 f(x^*) \prec 0$, then x^* is a local or global maximizer.*
- *If $\nabla^2 f(x^*)$ has negative and positive eigenvalues, then x^* is a saddle point.*

Figure 1-1 shows the possible critical points for a function in the domain $x \in [-3, 3]$.

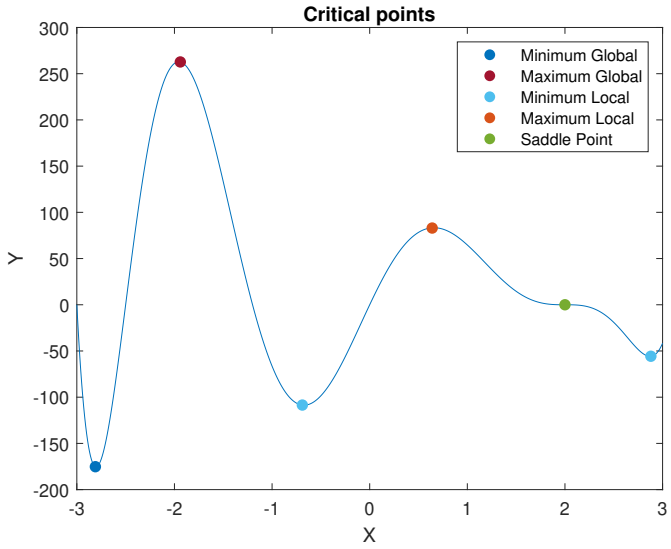


Figure 1-1: Visualization of critical points

1.2 Classical optimization Methods

This section will overview some of the main iterative methods to find a local minimizer of a smooth function f. These methods are the core of many current algorithms.

1.2.1 Gradient method

One of the most famous and well-studied algorithms is the gradient method. This algorithm is attributed to Augustin-Louis Cauchy with the paper [6] published in 1847. The gradient method is an iterative algorithm for $f \in \mathcal{C}^1$ that decreases the value function at each step. The idea is to follow the direction of the steepest slope, i.e. to take $-\nabla f(x_k)$ as the search direction.

$$x_{k+1} = x_k - \alpha_k \nabla f(x_k)$$

If the gradient is L-Lipschitz, the step size α_k can be chosen as $\alpha_k = \frac{1}{L}\ \forall\, k$. We recall the following definition of Lipschitz function continuity.

Definition 1.2.1 (Lipschitz continuity for gradient)**.** The gradient of the function f is said to be L-Lipschitz if,

$$\|\nabla f(x) - \nabla f(y)\| \leq L\|x - y\|\ \ \forall x, y \in \mathbb{R}^n$$

We will denote $f \in \mathcal{C}_L^{1,1}$ as a smooth function of degree 1 with its first derivative L-Lipschitz. The gradient method with this particular choice of step size has different convergence results. First, let's mention the global convergence theorem,

Theorem 1.7. *Let $f \in \mathcal{C}_L^{1,1}$, f bounded by f_{low} and x_0 the first iterate.*

$$\min_{i=0,\dots,k-1} \|\nabla f(x_i)\| \leq \sqrt{\frac{2L(f(x_0) - f_{low})}{k}}$$

This global convergence leads to the worst complexity bounds on the gradient norm regarding the number of iterations. Secondly, Nesterov [7, Theorem 1.2.4] shows the following local convergence result.

Theorem 1.8. *Suppose $f \in \mathcal{C}_M^{2,2}$ and,*

$$lI_n \leq \nabla^2 f(x) \leq LI_n$$

with $0 \leq l \leq L \leq \infty$. If we have $r_0 = \|x_0 - x^*\| \leq \bar{r} = \frac{2l}{M}$ then,

$$\|x_k - x^*\| \leq \frac{\bar{r} r_0}{\bar{r} - r_0}(1 - \frac{l}{L+l})^k$$

Therefore, the gradient method has a local linear convergence.

1.2.2 Newton's method

Newton's method of optimization is also an iterative method for finding a local optimum for $f \in \mathcal{C}^2$. The main difference with the gradient method is the use of the Hessian of the function to find a better next iterate. This method is based on minimizing the second-order Taylor approximation around x_k. This results in the following update of the iterates,

$$x_{k+1} = x_k - (\nabla^2 f(x_k))^{-1} \nabla f(x_k)$$

This equation is the use of Newton's Raphson method [8] published in 1690 to find the roots of a function. The problem is similar, as we can see the optimization of the function $f(x)$ by finding x such that $\nabla f(x) = 0$.

This method has a better convergence rate than the gradient method but is also more expensive. More precisely, Nesterov [7, Theorem 1.2.5] proved the following local convergence result.

Theorem 1.9. *Suppose $f \in \mathcal{C}_M^{2,2}$ and,*

$$lI_n \leq \nabla^2 f(x)$$

with $0 \leq l \leq \infty$. If we have $\|x_0 - x^*\| \leq \bar{r} = \frac{2l}{3M}$, then $\|x_k - x^*\| < \bar{r}$ $\forall\, k$ and

$$\|x_{k+1} - x^*\| \leq \frac{M\|x_k - x^*\|^2}{2(l - M\|x_k - x^*\|)}$$

This theorem ensures a quadratic convergence for Newton's method if we begin sufficiently close to the solution.

1.2.3 Quasi-Newton method

Despite the possible fast convergence of Newton's method, in practice, it may be impossible to use this method due to the computation of the Hessian. The family of quasi-Newton methods has been developed in this sense, trying to use second-order information without computing it. The main objective of this method is to approximate the Hessian with first-order information.

Therefore, the quasi-Newton method aims to find a good $H_k \approx \nabla^2 f(x_k)$.

$$x_{k+1} = x_k - \alpha_k H_k^{-1} \nabla f(x_k)$$

We want to update H_k at each iteration to improve the approximation of the Hessian. If we consider $f \in \mathcal{C}_M^{2,2}$ and $\|x_{k+1} - x_k\| \ll 1$ we have,

$$(\nabla^2 f(x_{k+1}))^{-1} s_k \approx y_k$$

with $s_k = x_{k+1} - x_k$ and $y_k = \nabla f(x_{k+1}) - \nabla f(x_k)$. This relation leads us to choose the approximation $B_k \approx (\nabla^2 f(x_{k+1}))^{-1}$ in such a way that

$$B_{k+1} s_k = y_k \qquad \text{(Secant condition)}$$

The Secant condition is a basis of many different quasi-Newton's methods. Working with B_k instead of H_k is a common choice that avoids the inverse in the computation of the next iterate. After that, there are many different ways to update B_k.

The first idea is to consider a symmetric rank-one (SR1) update given by,

$$B_{k+1} = B_k + uv^T$$

We update the matrix B_k with a rank-one perturbation from iteration to iteration. The vectors $u, v \in \mathbb{R}^n$ must be chosen to verify the Secant condition and preserve symmetry. A possible choice is given by,

$$v = s_k - B_k y_k \text{ and } u = \frac{v}{v^T y_k}$$

which gives the following update,

$$B_{k+1} = B_k + \frac{(s_k - B_k y_k)^T (s_k - B_k y_k)}{(s_k - B_k y_k)^T y_k}$$

One drawback of this method is that it doesn't ensure the positiveness of the matrix B_k. However, thanks to the second-order optimality conditions, we have seen that this was a requirement to have a minimizer. The second idea is to use this time a rank-two perturbation,

$$B_{k+1} = B_k + uu^T + vv^T$$

We can choose u and v such that B_{k+1} is a symmetric positive definite matrix if B_k is also. Moreover, we still have to respect the Secant condition. This method is called Broyden-Fletcher-Goldfarb-Shanno or BFGS.

$$B_{k+1} = (I - \frac{s_k y_k^T}{s_k^T y_k}) B_k (I - \frac{y_k s_k^T}{s_k^T y_k}) + \frac{s_k s_k^T}{s_k^T y_k}$$

The goal of developing these Quasi-Newton methods was to obtain a better convergence than the Gradient method without the computation cost of Newton's method. This goal is achieved; Quasi-Newton methods have a superlinear local convergence, as proved by Dennis and Moré [9].

Theorem 1.10. *Suppose that $f \in \mathcal{C}_M^{2,2}(\mathbb{R}^n)$ and B_k non singular $\forall\, k \geq 0$. We consider the iterative process,*

$$x_{k+1} = x_k - B_k \nabla f(x_k)$$

If the following conditions hold,

- *$\exists D \subset \mathbb{R}^n$ open and convex such that $\{x_k\}_{k\geq 0} \subset D$.*
- *$\lim_{k\to+\infty} x_k = x^*$, with $\nabla^2 f(x_k) \succeq \mu I_n$ for some $\mu > 0$.*
- *$\lim_{k\to+\infty} \dfrac{\|[B_k^{-1} - \nabla^2 f(x^*)](x_{k+1} - x_k)\|}{\|x_{k+1} - x_k\|} = 0$*

then $\nabla f(x^*) = 0$ and,

$$\lim_{k\to+\infty} \frac{\|x_{k+1} - x_k\|}{\|x_k - x^*\|} = 0$$

1.3 Monotone Line Search Methods

In this section, we will present an iterative idea to find the minimum of a function. Here, we will present the line search strategy, which consists of computing a search direction and step size to find the next iterations. Each line search method uses the following formula

$$x_{k+1} = x_k + \alpha_k d_k$$

where d_k is the direction and α_k the step size.

Some possible direction has already been discussed in the previous section. We can generalize the search directions as,

$$d_k = -B_k \nabla f(x_k)$$

where the choice of B_k expresses the method used. Indeed, taking $B_k = I\ \forall\, k$ gives the Gradient method when taking $B_k = (\nabla^2 f(x_k))^{-1}\ \forall\, k$ is the Newton's method. Moreover, choosing B_k as given by the SR1 or BFGS leads to a Quasi-Newton method.

Finally, the only difference between the monotone line search methods and the previous classical methods presented above lies in the step size α_k. His use makes sense in

different situations. For example, the optimal step size $\alpha_k = \frac{1}{L}$ proposed for the Gradient method may be impossible to compute in practice due to the Lipschitz constant. Moreover, the pure Newton's step states in the previous section are often replaced with a damped Newton's step in practice,

$$d_k = -\gamma(\nabla^2 f(x_k))^{-1}\nabla f(x_k) \quad \gamma \in (0,1)$$

We will present different possible choices for defining this step size α_k to verify some properties and have convergence results.

1.3.1 Armijo Line Search

The Armijo rule is used in the context of backtracking line search and was introduced by Armijo in [10]. The idea is to test smaller and smaller step sizes until a decrease in the objective function is sufficient. The Armijo rule can be used as a stopping criterion for this method and is given by the following condition.

$$f(x_k + \alpha_k d_k) \leq f(x_k) + c\alpha_k d_k^T \nabla f(x_k) \tag{1.2}$$

In this equation, d_k is the direction, α_k is the considered step size and $c \in (0,1)$. If the condition is not satisfied, we set $\alpha_k = \tau\alpha_k$ with $\tau \in (0,1)$ and check the condition until it is satisfied. Hopefully, this condition will always be satisfied in a finite time, as the following theorem states.

Theorem 1.11. *Let f be differentiable at $x \in \mathbb{R}^n$. If $\nabla f(x)^T d < 0$ and $\nu \in (0,1)$, then $\exists \delta > 0$ such that*

$$f(x + \alpha d) \leq f(x) + \nu\alpha\nabla f(x)^T d, \quad \forall\, \alpha \in [0, \delta).$$

Furthermore, the backtracking line search with the stopping criterion of the Armijo rule has some global convergence. Absil, Mahony & Andrews [11] have proved that this method always converges to a critical point if the function under consideration is a real analytic function. In short, a real analytic function can be written with convergent power series with real coefficients. For example, polynomials and trigonometric functions are real analytic. The critical points also include saddle points which are not interesting. It has been proved by Panageas, Piliouras & Wang [12] that if the gradient is L-Lipschitz and if we use specific methods like gradient descent, the algorithms will avoid strict saddle points.

Figure 1-2 shows the consequence of the Armijo conditions on an example function.

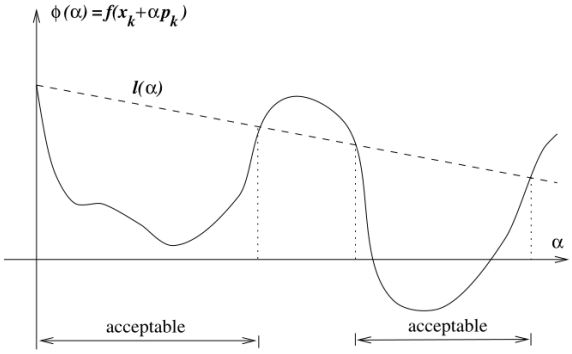


Figure 1-2: The acceptable steplength for the Armijo conditions from [13]

1.3.2 Other Monotone Line Searches

There exist other ways to define the step size for monotone line search methods; we will present here some of them. The description of the following linear search methods follows the page by Lihe Cao et al. [13].

The weak Wolfe conditions introduced by Philip Wolfe [14] will accept a step size α_k if the two following conditions are satisfied,

$$f(x_k + \alpha_k d_k) \leq f(x_k) + c_1 \alpha_k d_k^T \nabla f(x_k) \quad \text{(Armijo Condition)}$$
$$-d_k^T \nabla f(x_k + \alpha_k d_k) \leq -c_2 d_k^T \nabla f(x_k) \quad \text{(Curvature Condition)}$$

with $0 < c_1 < c_2 < 1$. If the second condition is replaced by the following, this is called the strong Wolfe condition.

$$|d_k^T \nabla f(x_k + \alpha_k d_k)| \leq c_2 |d_k^T \nabla f(x_k)| \quad \text{(Strong Curvature Condition)}$$

The existence of an α_k that satisfies these conditions is guaranteed if we use the gradient descent and the function is bounded below. Moreover, with the same two assumptions, the Zoutendijk condition proposed by G. Zoutendijk [15] holds,

$$\sum_{k=0}^{\infty} (\cos^2(\theta_k)) \|\nabla f(x_k)\|^2 \leq \infty$$

with $\cos(\theta_k) = \frac{-\nabla f(x_k)^T d_k}{\|\nabla f(x_k)\| \|d_k\|}$. This means that $(\cos^2(\theta_k))\|\nabla f(x_k)\|^2$ goes to 0 or that we reach a critical point when $\nabla f(x_k) \to 0$.

Finally, we state the Goldstein conditions, which look like the Wolfe conditions,

$$f(x_k + \alpha_k d_k) \leq f(x_k) + c\alpha_k \nabla f(x_k)^T d_k$$
$$f(x_k + \alpha_k d_k) \geq f(x_k) + (1 - c)\alpha_k \nabla f(x_k)^T d_k$$

with $c \in [0, \frac{1}{2}]$. These conditions give a range of acceptable function values below the

current one. Convergence property for Goldstein conditions can be described similarly to the Wolf conditions with the Zoutendijk condition.

We can see the influence of the curvature condition on 1-3 and the Goldstein condition on 1-4.

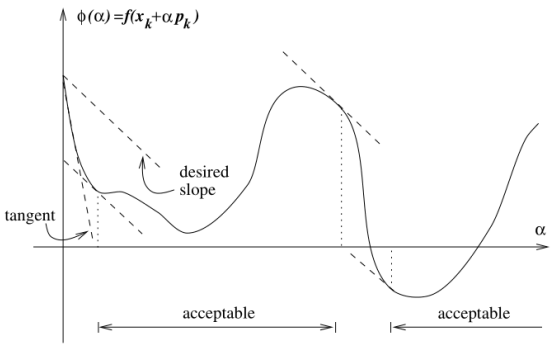


Figure 1-3: The acceptable steplength from the curvature conditions from [13]

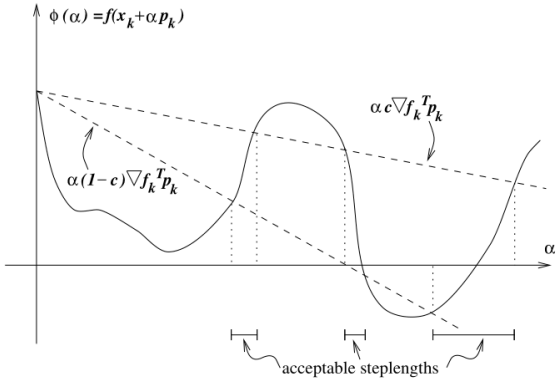


Figure 1-4: The acceptable steplength from the Goldstein conditions from [13]

1.4 Non-Monotone Line Search Methods

All the line search methods described above generate the sequence $\{x_k\}_{k\geq 0}$ to obtain a monotonically decreasing sequence $f(x_k)$. These techniques are efficient for convex problems because reaching a critical point gives a global minimum thanks to 1.5. Due to the existence of local minima or saddle points, these methods may struggle to find interesting solutions for non-convex problems. In fact, they quickly get stuck on the first critical point they find.

This observation is the main reason for our interest in the non-monotone methods. The idea is quite simple; we want to allow some iterations to increase the value of the function in the hope of escaping local minima. This non-monotonic behavior has to disappear with iterations to have a convergent algorithm.

1.4.1 Grippo-Lampariello-Lucidi method

One of the first non-monotone methods was proposed by Grippo, Lampariello and Lucidi [4]. They aimed to use non-monotonic behavior to overcome the drawbacks of Newton's methods. Indeed, Newton's method is known to diverge quickly if the starting points are not in the pool of attraction of the method, and it can also converge to a local maximum. This last drawback can be easily solved using $\alpha_k = sgn(d_k^T H_k^{-1} d_k)$, which ensures the use of descent directions. The line search technique can be used to ensure global convergence, for example, with the Armijo rule, which enforces a monotonic decrease. Unfortunately, this solution can considerably slow the Newton method's convergence. Therefore, Grippo, Lampariello and Lucidi proposed the following generalization of the Armijo rule.

$$f(x_k + \alpha_k d_k) \leq f_{l(k)} + \gamma \alpha_k \nabla f(x_k)^T d_k$$

with the factor $\gamma \in (0, 1)$, the step-size α_k and

$$f_{l(k)} = \max_{0 \leq j \leq m(k)} [f_{k-j}] \tag{1.3}$$

where $m(0) = 0$ and $0 \leq m(k) \leq \min[m(k-1) + 1, M]$ with M a non-negative integer. The idea was to allow bad choices at some iterations, thanks to the max term, while maintaining the global convergence of the Armijo rule. Although this method was designed to overcome Newton's method's drawback, allowing some non-monotonic behavior opens the door to general non-monotone line search methods.

1.4.2 Zhang and Hager

Thanks to the work of Grippo, Lampariello and Lucidi, other non-monotone line search methods have been developed. The one by Zhang and Hager [5] is based on it but with some modifications. First, they compute the descent direction thanks to a BFGS update. Second, they consider an average of previous function values to obtain non-monotonic behavior. The core of the proposed algorithm is as follows,

$$f(x_k + \alpha_k d_k) \leq C_k + \delta \alpha_k \nabla f(x_k)^T d_k$$

We recognise an Armijo line search where we have added this non-monotone term C_k which is defined recursively for a given choice of $\eta_k \in [\eta_{min}, \eta_{max}], 0 \leq \eta_{min} \leq \eta_{max} \leq 1$ by,

$$Q_{k+1} = \eta_k Q_k + 1 \qquad C_{k+1} = (\eta_k Q_k C_k + f(x_{k+1}))/Q_{k+1}$$

and $C_0 = f(x_0)$. The next part of the paper [5] focused on proving a certain convergence under different assumptions. The aim of this new method is quite different from that of Grippo, Lampariello and Lucidi. It was designed to solve non-convex problems better than other non-monotone schemes and is one of the first to use the line search technique in this sense.

1.4.3 Extensions

After the above two papers on the first non-monotone line search algorithms, many different combinations of these two algorithms have emerged. As mentioned in M. Ahookhosh et al. [16], many studies prove that using non-monotone strategies increases the chance of finding interesting solutions. The well-known desired behavior for these new algorithms is to have a strong non-monotone behavior far from the solution and a weaker one close to it. We will present three new non-monotone methods to improve the existing version.

The first is proposed by M. Ahookhosh et al. [16]. Their idea is to combine the non-monotone strategy with a modified Armijo rule. Their algorithm can be written as follows,

$$f(x_k + \alpha_k d_k) \leq D_k + \sigma\alpha_k \left[\nabla f(x_k)^T d_k + \gamma \|\nabla f(x_k)\|^2\right].$$

The first change is the use of an adjustable initial step length α_k, defined thanks to the exact one if the function $f(x)$ were quadratic convex, i.e.

$$\alpha_k = -\frac{\nabla f(x_k)^T d_k}{d_k^T B_k d_k}$$

where B_k is an approximation of the Hessian. After that, they use the value D_k, which is defined as,

$$D_k = \begin{cases} f(x_0) & \text{if } k = 0 \\ f(x_k) + \eta_{k-1}(D_{k-1} - f(x_k)) & \text{else} \end{cases}$$

with $\eta_{k-1} \in [\eta_{min}, \eta_{max}]$ for $\eta_{min} \in [0, 1)$ and $\eta_{max} \in [\eta_{min}, 1)$.

The second one we present here is described in M. Ahookhosh et al. [17]. It is a variation of the one proposed by Grippo et al. [4], where they try to improve the use of the previous function values. The algorithm [4] uses a maximum that eliminates all the influence of the interesting function values already found. M. Ahookhosh et al. [17] proposed,

$$f(x_k + \alpha_k d_k) \leq R_k + \delta\alpha_k \nabla f(x_k)^T d_k$$

where R_k is defined as,

$$R_k = \eta_k f_{l(k)} + (1 - \eta_k) f_k$$

with $0 \leq \eta_{min} \leq \eta_{max} \leq 1$ and $\eta_k \in [\eta_{min}, \eta_{max}]$. We see in the term R_k the non-monotonous term $f_{l(k)}$ given in 1.3 combined with the previously obtained function values.

The last example of a new non-monotone algorithm we will see is proposed by M. Ahookhosh and S. Ghaderi [18]. They want to consider the convex combinations of only the N previous function values. The algorithm is as usual,

$$f(x_k + \alpha_k d_k) \leq T_k + \delta\alpha_k \nabla f(x_k)^T d_k$$

where T_k is defined as,

$$T_k = \begin{cases} f_{l(k)} & \text{if } k \leq N \\ \max\{\bar{T}_k, f_k\} & \text{else} \end{cases}$$

This term $\bar{T}_k$ is computed recursively with,

$$\begin{aligned} \bar{T}_0 &= f_0 \\ \bar{T}_k &= (1 - \eta_{k-1}) f_k + \eta_{k-1} \bar{T}_{k-1} + \xi_k (f_{k-N} - f_{k-N-1}) \end{aligned}$$

Since the goal is to get a convex combination, ξ_k is defined so that the sum of the parameters in front of the past value functions is 1. Therefore, we must have

$$\xi_k = \eta_{k-1}\eta_{k-2}...\eta_{k-N-1}$$

which is null if $k < N$.

1.5 General Framework for Non-Monotone Line Searches

With all these new methods, a convergence analysis of these types of algorithms has been made by various authors. The work of Cartis, Sampaio and Toint in [19] focuses on non-monotone gradient-related line search optimization methods based on methods of the form proposed by Grippo et al. [4]. They show that the worst-case complexity is also $\mathcal{O}(\epsilon^{-2})$, just as the gradient method.

As we can see, there was a need to generalize the results. This work was done by Sachs and Sachs [20]. They aimed to develop a general convergence theory for all these new non-monotone line search methods developed in the context of global optimization problems. A formulation of the various non-monotone algorithms above and others in a generic algorithm was proposed by Grapiglia and Sachs [1]. The results were completed and even further generalized in Grapiglia and Sachs [2], where they used the following extension of the scheme proposed by Sachs and Sachs [20].

Algorithm 1 General non-monotone descent algorithm [2]

1: **Step 0** Given $x_0 \in \mathbb{R}^n$, $\alpha_0 > 0$ and $\beta, \rho \in (0,1)$, set $k := 0$.
2: **Step 1** Compute a descent direction $d_k \in \mathbb{R}^n$ for x_k.
3: **Step 2.1** Set $l := 0$.
4: **Step 2.2** Choose $v_{k,l} \geq 0$. If

$$f(x_k + \alpha_k\beta^l d_k) \leq f(x_k) + \rho\alpha_k\beta^l \nabla f(x_k)^T d_k + v_{k,l} \tag{1.4}$$

set $l_k = l$ and $v_k = v_{k,l_k}$ and go to Step 3. Otherwise, set $l := l + 1$ and repeat Step 2.2.
5: **Step 3** Set $x_{k+1} = x_k + \alpha_k\beta^{l_k} d_k$, $\alpha_{k+1} = \alpha_k\beta^{l_k - 1}$, $k := k + 1$ and go to Step 1.

This algorithm is based on an Armijo line search where the term $v_{k,l}$ has been added and is responsible for the non-monotonicity. The modification by Grapiglia and Sachs [2] implies the term $v_{k,l}$ by allowing it to differ as a function of l. This difference extends the possibilities.

Remark 1.1. In this paper, we will always use the notation v_k which refers to $v_k = v_{k,l_k}$ in Algorithm 1 and not to the v_k introduced by Sachs and Sachs [20].

Grapiglia and Sachs [1, 2] have covered the worst-case complexity analysis of this algorithm. With the appropriate assumptions, this general algorithm reaches the

stopping criterion $\|\nabla f(x_k)\| \leq \epsilon$ in $\mathcal{O}(\epsilon^{-2})$ iterations for potentially non-convex functions. The result improves when we deal with convex functions by reaching $f(x_k) - f(x^*) \leq \epsilon$ in $\mathcal{O}(\epsilon^{-1})$ iterations. To obtain these results, the following assumptions are required :

A1 The objective function $f : \mathbb{R}^n \to R$ is differentiable and its gradient $\nabla f : \mathbb{R}^n \to \mathbb{R}^n$ is Lipschitz continuous with Lipschitz constant $L > 0$.

A2 For all k,

$$\nabla f(x_k)^T d_k \leq -c_1 \|\nabla f(x_k)\|^2 \text{ and } \|d_k\| \leq c_2 \|\nabla f(x_k)\|$$

for some constants $c_1, c_2 > 0$.

A3 There exists $f_{low} \in \mathbb{R}$ such that $f(x) \geq f_{low}\ \forall x \in \mathbb{R}^n$.

A4 $\lim_{T \to +\infty} \frac{1}{T} \sum_{k=0}^{T-1} v_k = 0$

Most of the algorithms we will see have the sequence $\{v_k\}_{k \geq 0}$ summable (i.e. $\sum_{k=0}^{+\infty} v_k < +\infty$). Replacing the assumption **A4** above by this property, we can also find the complexity result in $\mathcal{O}(\epsilon^{-2})$. This work is already covered in [1]. We will also work with an algorithm that doesn't respect this property but does the **A4** assumption.

Moreover, it can be shown that $\sum_{k=0}^{+\infty} v_k < +\infty \implies$ **A4** as,

$$\lim_{T \to +\infty} \frac{1}{T} \sum_{k=0}^{T-1} v_k = \lim_{T \to +\infty} \frac{M}{T} = 0$$

with M the sum of the series which exists by assumption.

1.5.1 Worst case complexity analysis

We will briefly present the theorems that prove the convergence of all the methods that respect Algorithm 1. All these theoretical results come from Grapiglia and Sachs [2].

Lemma 1.12. *Suppose that **A1** and **A3** hold. Then, for all k we have,*

$$\alpha_k \geq \min \left\{ \alpha_0, \frac{2(1-\rho)c_1}{L c_2^2} \right\} \equiv \bar{\alpha}$$

Taken from [2, Lemma 2]

This lower bound on the step size is necessary to find an upper bound on the total number of iterations of Algorithm 1 for a certain k, this is done in [2, Theorem 1]. We will also use it for the following definition.

Definition 1.5.1. Given $\epsilon > 0$, we define $T_0(\epsilon)$ as any non-negative integer that respect the following condition,

$$T \geq T_0(\epsilon) \implies \frac{1}{T} \sum_{k=0}^{T-1} v_k \leq \frac{\kappa_c \epsilon^2}{2}, \quad \kappa_c = \rho \beta c_1 \bar{\alpha}$$

We are now ready to state the following theorem which is central for the complexity result of Algorithm 1.

Theorem 1.13. *Suppose that* ***A1-A4*** *hold and let the sequence* $\{x_k\}_{k\geq 0}$ *be generated by Algorithm 1. If,*

$$T \geq \max\left\{T_0(\epsilon), \frac{2(f(x_0) - f_{low})}{\kappa_c \epsilon^2}\right\} \tag{1.5}$$

then,

$$\min_{k=0,\dots,T-1} \|\nabla f(x_k)\| \leq \epsilon$$

Taken from [2, Theorem 2]

The theorem 1.13 will lead to two corollaries, one showing the complexity for summable series and the other for those who only respect assumption **A4**.

Corollary 1.1. *Suppose that* ***A1-A3*** *hold and that* $\sum_{k=0}^{+\infty} v_k < +\infty$. *Let* $\{x_k\}_{k=0}^{+\infty}$ *be a sequence generated by Algorithm 1. Given* $\epsilon \in (0,1)$, *if:*

$$T \geq 2\max\left\{\sum_{k=0}^{+\infty} v_k, f(x_0) - f_{low}\right\} \kappa_c^{-1} \epsilon^{-2} \tag{1.6}$$

then,

$$\min_{k=0,\dots,T-1} \|\nabla f(x_k)\| \leq \epsilon$$

Taken from [2, Corollary 1]

The results we obtained here are the same as the one proved in [1]. If **A1-A3** hold, $\sum_{k=0}^{+\infty} v_k < +\infty$ and the assumption (1.6) of Corollary 1.1 is respected, then the worst-case complexity bound is in $\mathcal{O}(\epsilon^{-2})$ iterations. We have worked differently as it will also allow us to find complexity bounds even if $\sum_{k=0}^{+\infty} v_k < +\infty$ is not respected, the following corollary gives this.

Corollary 1.2. *Suppose that* ***A1-A3*** *hold and that* $v_k \longrightarrow 0$. *Let constant* $C > 0$ *such that* $v_k \leq C \ \forall\, k$ *and, given* $\delta > 0$, *let* $k_0(\delta)$ *be a positive integer such that* $v_k \leq \delta$ *if* $k \geq k_0(\delta)$. *If,*

$$T \geq \max\left\{\frac{2k_0(\delta)C}{\delta}, 1 + k_0(\delta), \frac{2(f(x_0) - f_{low})}{\kappa_c \epsilon^2}\right\} \tag{1.7}$$

Then, for any sequence $\{x_k\}_{k=0}^{+\infty}$ *generated by Algorithm 1 and* $\delta = \frac{\kappa_c \epsilon^2}{2}$,

$$\min_{k=0,\dots,T-1} \|\nabla f(x_k)\| \leq \epsilon$$

Taken from [2, Corollary 2]

The development proposed in [2] is the basis to investigate other non-monotone terms v_k as $v_k = o(\|\nabla f(x_k)\|^2)$ with $k \longrightarrow 0$ for example. We can also generalize the results for many other non-monotone algorithms based on a modified Armijo rule as the methods by Grippo et al. [4] and Zhang and Hager [5] already seen.

1.5.2 The different methods

We consider here three different methods written in the form proposed by Algorithm 1.

Grippo, Lampariello and Lucidi [4] method :
This method presented in the last section can be written as Algorithm 1. The definition of $v_{k,l}$ is slightly modified compared to the source to stick to the generic algorithm.

$$v_{k,l} = \max_{0 \leq j \leq m_k} [f(x_{k-j})] - f(x_k) \ \forall k, l$$

To understand better the idea of this algorithm, let's take a look at the term $\max_{0 \leq j \leq m_k}[f(x_{k-j})]$. We define $x_k^* = \arg\max_{0 \leq j \leq m_k}[f(x_{k-j})]$ and observe that the sequence $f(x_k^*)$ is non-increasing. Indeed, we have,

$$f(x_{k+1}^*) = \max_{0 \leq j \leq m(k+1)}[f(x_{k+1-j})] \leq \max_{0 \leq j \leq m(k)+1}[f(x_{k+1-j})] = \max[f(x_k^*), f(x_{k+1})] = f(x_k^*)$$

The last equality comes from the update inequality as the scalar product is negative.

$$f(x_{k+1}) \leq f(x_k^*) + \rho \alpha_k \beta^l \nabla f(x_k)^T d_k$$

This form shows that the update inequality of NM1 is similar to the classical Armijo line search. Intuitively, we can try to understand the impact of using $f(x_k^*)$ instead of $f(x_k)$. As $f(x_k^*) \geq f(x_k)$, the update step will allow us to not fall too quickly on a critical point by choosing a worse value function. However, as the sequence $f(x_k^*)$ is non-increasing, this behavior will slowly disappear, and the algorithm will converge. Moreover, we can control the number of iterations affected by an old poor value function with the factor M. We expect that with bigger M, the algorithm may take more iterations to converge but to a better critical point.

The Zhang and Hager [5] method:
This method can again be described in the form of Algorithm 1, where the non-monotone term $v_{k,l}$ is defined as,

$$v_{k,l} = C_k - f(x_k) \ \forall k, l$$

The idea of this algorithm, as presented in the previous chapter, is to replace the classical Armijo line search with a convex combination of the previous function values. This convex combination is determined by the parameter η_k, if its value is 0, then we obtain the Armijo line search. We get the balanced average function value by fixing it

to 1. Therefore, the bigger η_k is, the more the non-monotonic nature of the algorithm will increase.

The metropolis-based non-monotone method by Grapiglia and Sachs [2]:

This method is proposed by Grapiglia and Sachs [2] and is based on the metropolis-based non-monotone algorithm. As a reminder, the metropolis rule works as follows with a uniform random number $p_k \in [0, 1]$,

$$x_{k+1} = \begin{cases} x_k^+ & \text{if } p_k \leq \min\left\{1, \exp\left(-\frac{f(x_k^+)-f(x_k)}{\tau_k}\right)\right\} \\ x_k & \text{else.} \end{cases}$$

The goal, in this case, is to allow x_{k+1} to be different from x_k randomly. We can adapt this idea to our algorithm by letting $x_{k+1} = x_k^+$ even if $f(x_k^+) > f(x_k)$. Moreover, this event must arrive with a probability inversely proportional to the difference $f(x_k^+) - f(x_k)$. Finally, this non-monotone behavior must become unlikely with the number of iterations. All these features are already present in the Metropolis rule. Therefore, Grapiglia and Sachs [2] defined the non-monotone term as,

$$v_{k,l} = \sigma \exp\left(-\frac{\max(\theta, f(x_k + \alpha_k \beta^l d_k) - f(x_k))}{\tau_k}\right)$$

where,

$$\tau_k = \frac{1}{\ln(k+1)}$$

The idea of NM3 is to use both parameters σ and θ to impact the algorithm's non-monotonicity. The bigger σ is, the bigger $v_{k,l}$ will be, allowing more poor iterates to be chosen. The smaller θ is, the bigger $v_{k,l}$ will be. Therefore, both parameters will allow the algorithm to escape from local minimizers. The value τ_k allows us to choose the number of iterations which will be impacted by the non-monotone behavior.

Thanks to the worst-case complexity analysis of Algorithm 1, we can prove the complexity of this method. This part is detailed in [2, Section 4].

The results must be separated into $\theta > 1$ and $\theta \in (0, 1]$. They will use Corollary 1.1 and Corollary 1.2, respectively, to find a bound.

Theorem 1.14. *Suppose that* ***A1-A3*** *hold and let the sequence* $\{x_k\}_{k=0}^{+\infty}$ *be generated by Algorithm 1. Given* $\epsilon > 0$*, if* $\theta > 1$ *and,*

$$T \geq 2 \max \left\{ \sigma \sum_{k=0}^{+\infty} \frac{1}{(k+1)^{\theta}}, f(x_0) - f_{low} \right\} \kappa_c^{-1} \epsilon^{-2} \tag{1.8}$$

then,

$$\min_{k=0,\dots,T-1} \|\nabla f(x_k)\| \leq \epsilon$$

Taken from [2, Theorem 5]

Theorem 1.15. *Suppose that* ***A1-A3*** *hold and let the sequence* $\{x_k\}_{k=0}^{+\infty}$ *be generated by Algorithm 1. Given* $\epsilon > 0$*, if* $\theta \in (0, 1]$ *and,*

$$T \geq 2 \max \left\{ \left(\frac{4}{\kappa_c}\right)^{\frac{1+\theta}{\theta}} \sigma^{\frac{1}{\theta}}, 1 + \left(\frac{4\sigma}{\kappa_c}\right)^{\frac{1}{\theta}}, \frac{2(f(x_0) - f_{low})}{\kappa_c} \right\} \epsilon^{\frac{-2(1+\theta)}{\theta}} \tag{1.9}$$

then,

$$\min_{k=0,\dots,T-1} \|\nabla f(x_k)\| \leq \epsilon$$

Taken from [2, Theorem 6]

These two theorems give an upper bound on the number of function evaluations needed to reach a local optimum. We have,

$$T = \arg\min k \in \mathbb{N} : \|\nabla f(x_k)\| \leq \epsilon$$

which is the smallest number of iterations that satisfies the condition on the gradient. The assumptions given by the equations (1.8) and (1.9) are the upper bound on T. By contradiction, suppose that $T \geq$(1.8) or $T \geq$(1.9) then the theorem 1.14 or theorem 1.15 holds respectively. So we have,

$$\min_{k=0,\dots,T-1} \|\nabla f(x_k)\| \leq \epsilon$$

which contradicts the definition of T given above. Indeed, T is not the minimum integer that satisfies the condition, as $T - 1$ does.

The worst case complexity find for this algorithm for $\theta > 1$ and $\theta \in (0, 1]$ is in $\mathcal{O}(\epsilon^{-2})$ and $\mathcal{O}\left(\epsilon^{-\frac{2(1+\theta)}{\theta}}\right)$ respectively. The gap between these two cases at $\theta = 1$ is unexpected. It should probably be possible to find a better bound for the case $\theta \in (0, 1]$ to have a continuous behavior in the complexity.

1.6 Monotone vs Non-Monotone

Let's introduce a simple example of the behavior of the monotone and non-monotone line search algorithms for a global optimization problem. We show on the graph 1-5 the results of one of these algorithms on an interesting function which will be introduced later.

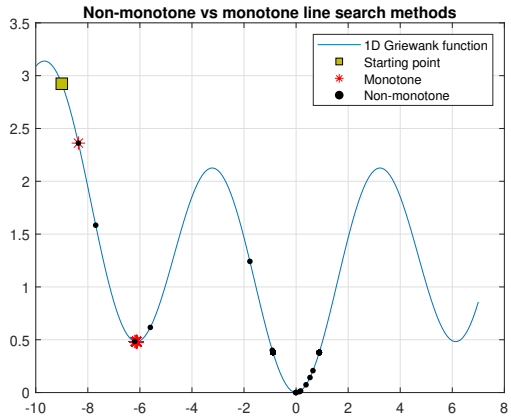


Figure 1-5: Numerical demonstration of the non-monotone behavior.

This graph rekindles the interest in exploring non-monotone line search algorithms further. Both monotone and non-monotone algorithms start from the starting point shown in the figure. After only two iterations, the monotone iterations sequence $\{x_k\}_{k\geq 0}$, represented by the red stars, reaches a local minimum with some precision. Since the function values generated by the sequence $\{x_k\}_{k\geq 0}$ must decrease monotonically, the monotone algorithm gets stuck in this local optimum.

If we focus on the non-monotone algorithm, represented by the black dots, we notice that it starts with the same iteration. After exploring the local minima, adding the non-monotone term $v_{k,l}$ introduced earlier allowed it to escape and find the global minima at 0.

We can see from this state-of-the-art that the field of non-monotone line search methods for solving non-convex problems is relatively young. The first proposals of algorithms are around 1986 and 2004 for Grippo et al. [4] and Zhang and Hager [5], respectively. The bound search by Cartis, Sampaio and Toint are published in 2014. Finally, the generalization work of Grapiglia and Sachs [1, 2] is presented in 2017 and 2019.

As these non-monotone line search methods are young, they still need to be proven. That's why there is a place for numerical experiments and comparison with a set of problems between these algorithms. This step is essential to show the added value of the non-monotone algorithms over the monotone ones. Validation and recognition are essential for all new mathematics and other development fields.

Moreover, there is a place for these algorithms to compete with other methods for non-convex problems, such as the famous Simulated Annealing algorithm. The pros

and cons of using local search for non-convex functions remain to be analyzed for these new algorithms.

Finally, some interesting real-world problems may be solved with a non-monotone line search method. In fact, for some of them, we can use additional information as the optimal function value, as in the well-known Euclidean distance geometry problem. This can be investigated further, and we will do so.

Chapter 2

Numerical investigation

This chapter presents the results of extensive numerical experiments with monotone and non-monotone line-search methods applied to benchmark global optimization problems. We first introduce different algorithms and the expected behavior on some preliminary numerical experiments. Then, we state the test problems, the methods' hyperparameters and the different comparison tools. This allows us to compare the non-monotone methods with the monotone method. Finally, additional comparisons are made with the well-known Simulated Annealing algorithm.

2.1 Implementations

This section will state the four methods considered in Grapiglia and Sachs [2]. The first one is the monotone line search when the three after are already presented in the State-of-the-Art. They all respect the generic scheme of Algorithm 1 with a particular choice for $v_{k,l}$. Therefore, the goal is to compare its impact on the solution found, knowing that it is responsible for the non-monotonicity of the method.

To make the comparison relevant, the four methods similarly define the descent direction d_k and the constants α_0, β and ρ. Grapiglia and Sachs [2] propose the following definitions :

- The constant values $\alpha_0 = 1$ and $\beta = \rho = 0.5$.
- The descent direction $d_k = -B_k \nabla f(x_k)$ where B_k is defined with the BFGS update rule :

$$B_{k+1} = \begin{cases} (I - \frac{s_k y_k^T}{s_k^T y_k}) B_k (I - \frac{y_k s_k^T}{s_k^T y_k}) + \frac{s_k s_k^T}{s_k^T y_k} & \text{if } s_k^T y_k > 0 \\ B_k & \text{else,} \end{cases}$$

  where $B_0 = I, s_k = x_{k+1} - x_k$ and $y_k = \nabla f(x_{k+1}) - \nabla f(x_k)$.

Note that the definition of the BFGS rule differs slightly from the previous chapter's definition. We have added a condition on the positivity of $s_k^T y_k > 0$, which is directly related to the positivity of B_k. With a constant unitary step size, theory shows that B_{k+1} is positive definite if B_k is positive definite. This property is no longer actual with an Armijo line search algorithm. Therefore, we use a safeguard when updating the matrix B_k.

The different methods are listed below:

- The first algorithm proposed by Grapiglia and Sachs [2] follows Algorithm 1 and uses $v_{k,l} = 0 \ \forall k, l$. This algorithm is referred to as "M1".
- The algorithm from Grippo et al. [4] presented in chapter 1 which follow Algorithm 1. This algorithm is referred to as "NM1". The value of M used in [2] for this method is 10. Moreover, [4] proposed to begin with classical Armijo line search and switch on the update of NM1 after M iterations. Intuitively, this prevents the algorithm from being delayed by a too-poor initial value function.
- The algorithm from Zhang and Hager [5] presented in chapter 1 which follow Algorithm 1. This algorithm is referred to as "NM2". Grapiglia and Sachs [2] used $\eta_{k-1} = \frac{0.85}{k}$. This choice allows the algorithm to have a strong non-monotone behavior on the first iterations and goes to a monotone behavior as the number of iterations increases.
- The algorithm from Grapiglia and Sachs [2] presented in chapter 1 which follow Algorithm 1. This algorithm is referred to as "NM3". The values for σ and θ will be discussed further in this chapter.

2.2 Preliminary Numerical Results

To validate the four algorithms described above, we will try them on a classical test problem for global optimization.

The aim is to reproduce the results obtained with the Griewank function selected by [5]. This function contains only one global minimum at $f(0, 0) = 0$ and numerous local minima, it is given by,

$$f(x) = 1 + \frac{x_1^2}{4000} + \frac{x_2^2}{4000} - \cos(x_1)\cos\left(\frac{x_2}{\sqrt{2}}\right)$$

To understand the challenge for our algorithm, let's pay attention to the graphs 2-1.

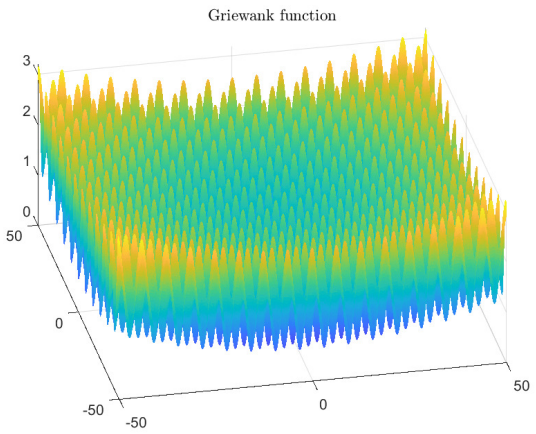


Figure 2-1: The valley of the GW function

On the view of the function, a monotone algorithm will fall directly into one of the numerous local minima. We will investigate if our non-monotone methods tackle this behavior.

As proposed in [5], we will consider 60 points in the box $[-600, 600]^2$ distributed like this :

$$\left(-600 + \frac{1200(i-1)}{3}, -600 + \frac{1200(j-1)}{14}\right), i \in 1, ..., 4 \text{ and } j \in 1, ..., 15$$

We have used the maximum number of iterations $k \leq 500$ and kept the best values found. This work's numerical experiments have been done on a PC with a 2.4 GHz Intel(R) i9-10885H processor and a R2019a version of Matlab. The results can be seen on the graph 2-2.

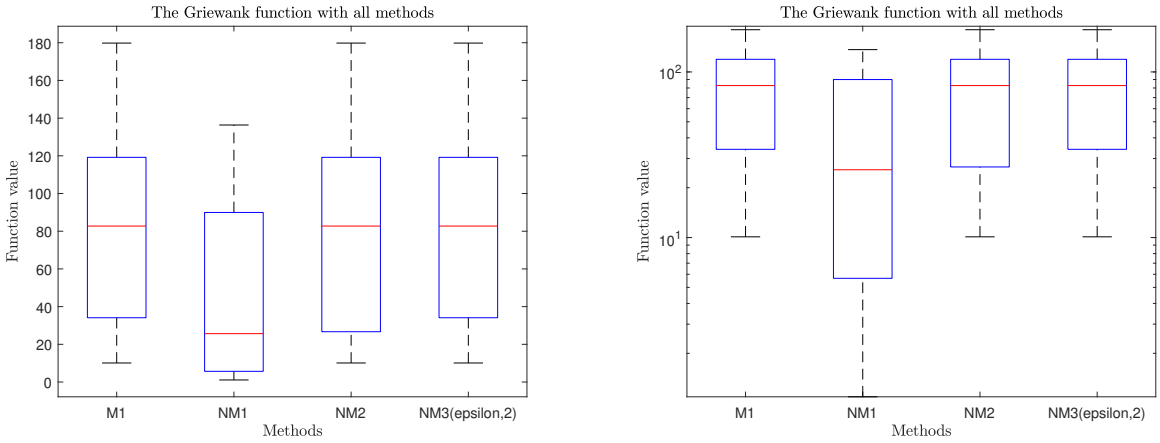


Figure 2-2: Box plots for all the methods on the GW function with 500 iterations

The box plots 2-2 presented here claim identical results than [5, Table. 2]. The methods NM2 and NM3, despite their non-monotone behavior, are unable to get out of a local minimum with the parameters given as they give precisely the same result for all the points evaluated as the monotone algorithm M1.

On the contrary, NM1 succeeds in escaping from some local minima and finding relevant solutions.

Let's look into the method NM3, which is supposed to be non-monotone. In the description of the metropolis-based non-monotone algorithm, we have said that increasing σ and decreasing θ will improve the non-monotonicity of the method. Therefore, we will follow the analysis of Grapiglia and Sachs [2] by considering $\sigma = |f(x_0)|$ and $\theta = \{4, 2, 1, 0.5, 0.25, 0.125\}$ allowing to each method $k_{\max} = 500$ iterations.

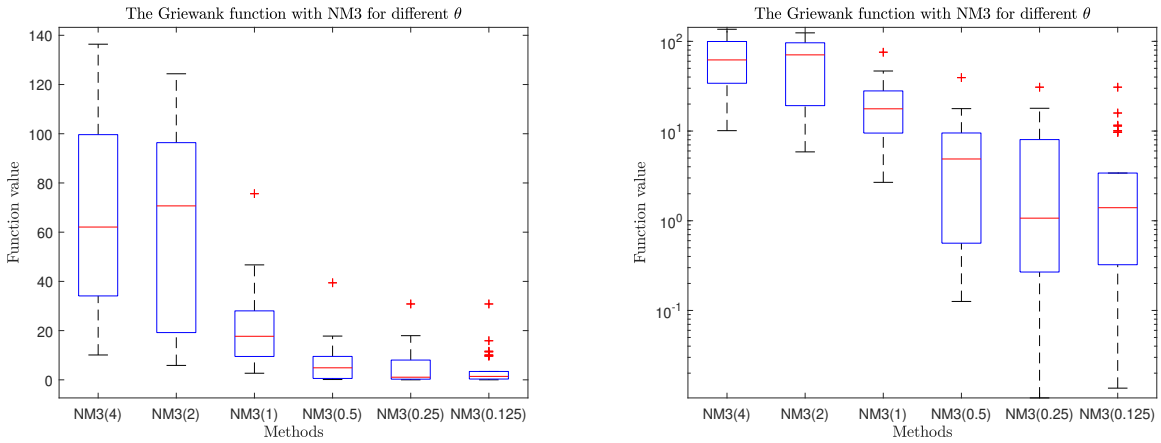


Figure 2-3: Box plots for NM3 with different θ on the GW function with 500 iterations

As we can see on figure 2-3, NM3 gets really interesting results with $\sigma = |f(x_0)|$ and low values for θ. With a minimum of 0.0106 with $\theta = 0.25$, NM3 is competitive against the other methods. The box plots 2-3 give the same information than [2, Table. 3].

Before going on to analyze other problems, we can compare the performance of this new method NM3 with its parent method, the Simulated Annealing (SA). This algorithm is also used on non-convex global optimization problems. This method doesn't use gradient descent and is based on heuristics with limited convergence guarantees. A significant advantage of Simulated Annealing is that it never gets stuck in a local optimum as it always continues to explore the function. Like NM3, Simulated Annealing is based on the Metropolis rule, which makes the comparison even more interesting. We will compare our algorithms with Matlab's version, *simulannealbnd*. To get a fair comparison, we will set the maximum number of function evaluations to 300.

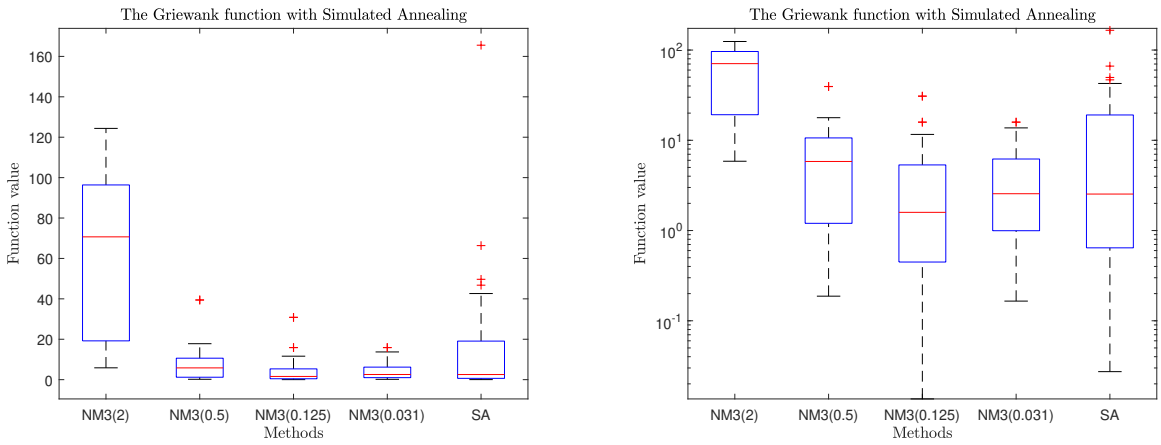


Figure 2-4: Comparison of NM3 and the SA on the GW function

The results in figure 2-4 show the competitiveness of the NM3 method compared to Simulated Annealing. With $\theta = 0.125$, NM3 manages to find better values than SA. It also achieves better average values and leads to worse values than SA.

2.3 Numerical Comparison on Benchmark Problems

This section will present a transparent comparison between the different methods and their results. We will describe the test problems defined by the test functions and the starting points. Some tools to compare the methods, such as the data and performance profiles, will be presented. Finally, after tuning the parameter θ in NM3, we will present the main results of comparing the monotone and non-monotone methods. An additional comparison with Simulated Annealing is proposed at the end.

2.3.1 Test Problems

First of all, the selected test functions must respect the following criterion.

- The function must be differentiable everywhere. All methods use a first-order method to find the search direction.
- We aim to show the ability of non-monotone methods to escape local minima. In this sense, we want test functions for which the number of local minima is either unknown or grows exponentially with the problem dimension

Considering these two criteria, we have chosen 20 problems from Ali, Khompatraporn and Zabinsky [21]. The subset of test functions chosen is described in annex A.

The choice of starting points for the comparison must try to explore the domain of each function as much as possible. Moreover, their construction must consider the domain's dimension and its boundary. Finally, we must be able to choose the same number of starting points for each problem, whatever their dimension, to obtain a fair comparison. All these criteria are met by the following generic way of defining starting points.

$$S_n(p) = \bigcup_{j=1}^{p} \left\{ \hat{x} + j\frac{b-a}{2p} d : d \in \mathcal{D}_n \right\}$$

with

$$\mathcal{D}_n = \{\pm e_i : i = 1, ..., n\}$$

and n, the dimension of the problem, a and b the lower and upper bound, $\hat{x}$ the center of the domain and $p = \frac{M}{2n}$ with M the least common multiple of $2n$ for the problems we have.

The behavior of this rule can be seen on the graph 2-5 in 2 dimensions for a box $[0, 10]^2$.

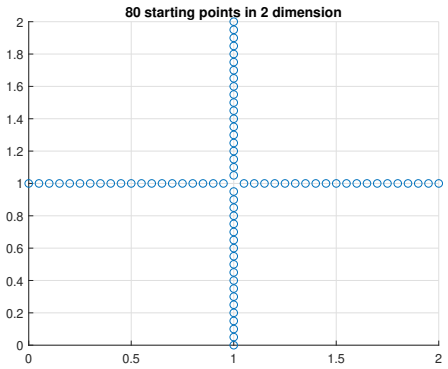


Figure 2-5: Example of the general generating points procedure

The following numerical comparison will refer to a test problem as a pair (function, starting point). We have $M = 360$ and, therefore, 360 starting points for each test function. This results in 7200 test problems.

2.3.2 Data and Performance Profiles

Comparing different methods with 7200 test problems in one graph can be perilous. This is why we introduce two different ways proposed in Moré and Wild [22].

The data profile

The data profile focuses on the number of problems that can be solved with a degree of precision and a specific number of function evaluations. Therefore, it is particularly interesting if the function evaluation is costly.

We first define the desired accuracy with,

$$f(x) \leq f_L + \tau(f(x_0) - f_L) \tag{2.1}$$

where f_L is the best value found for all the solvers on a problem p.

The data profile uses the following definition for a solver (method) s,

$$d_s(\alpha) = \frac{1}{|\mathcal{P}|} \left\{ p \in \mathcal{P} : \frac{t_{p,s}}{n_p + 1} \leq \alpha \right\}.$$

Where α is the number of function calls, $\mathcal{P}$ represents the set of test problems, and $t_{p,s}$ is the number of function evaluations required to achieve the accuracy (2.1) for all test problems p and solvers s. Finally, $d_s(\alpha)$ gives the number of problems solved in the sense of (2.1) by the solver s with a budget of α function evaluations. This definition allows us to get a fair comparison even for problems with different dimensions since we use the scaling factor $n_p + 1$ where n_p is the dimension of the test problem p.

The performance profile

Another tool available to compare different algorithms is the performance profile. This new profile introduces a performance ratio called α, which gives information about the additional iterations needed to achieve a given accuracy. The performance profile is defined as,

$$\rho_s(\alpha) = \frac{1}{|\mathcal{P}|}\text{size}\{p \in \mathcal{P} : r_{p,s} \leq \alpha\}$$

with

$$r_{p,s} = \frac{t_{p,s}}{\min\{t_{p,s} : s \in \mathcal{S}\}}$$

The performance measure $t_{p,s}$ is defined as before by the number of function evaluations required to pass the accuracy test (2.1). Thanks to the definition of the performance profile, we can obtain the number of problems for which the solver s is the best with $\rho_s(1)$. In addition, $\rho_s(2)$ shows the number of problems where the solver was at most twice as slow as the others. However, the cost of this new knowledge is the loss of information about the number of function evaluations required to solve a certain percentage of problems. This information can be found in the data profile.

The parameter α, which gives the number of function evaluations allowed, remains to be chosen for both data and performance profiles. A common choice proposed by Moré and Wild [22] is the α simplex gradient estimates, $\alpha(n_p + 1)$ with $\alpha = 100$. This choice allows more iterations for high-dimensional test problems.

2.3.3 Tuning parameter θ in NM3

We have seen in the preliminary numerical experiments that the hyperparameter choice can influence the NM3 method's performance. Therefore, looking for a relevant parameter for the new test problems is important. This problem can be seen as the following optimization problem,

$$\begin{aligned} &\max_{\theta} \; d_{NM3(100)} \\ &\text{s.t.} \quad \theta \geq 0 \end{aligned}$$

where $d_{NM3(100)}$ is defined by the data profile as the percentage of problems solved using $\alpha = 100$.

To solve this optimization problem, we will use the *fminsearch* function of Matlab for different initial guesses.

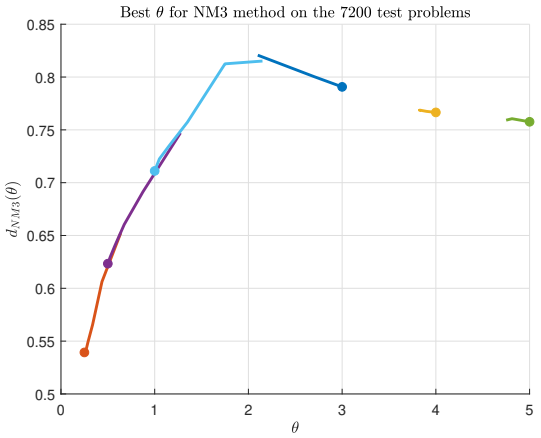


Figure 2-6: The factor $d_{NM3(100)}$ in function of θ for the 7200 test problems

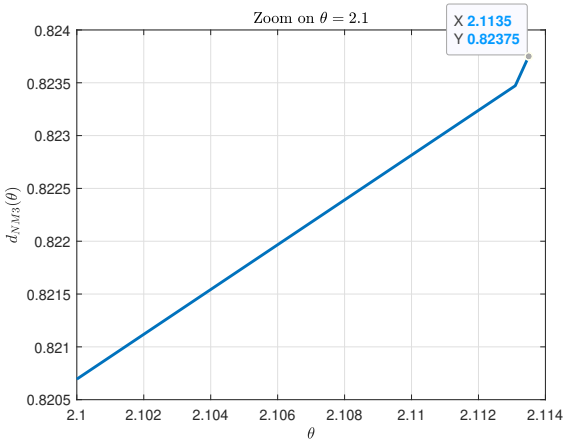


Figure 2-7: The factor $d_{NM3(100)}$ in function of θ for the 7200 test problems zoomed

We have chosen the set $[0.25, 0.5, 1, 3, 4, 5]$ as initial θ for the *fminsearch* function. The graph 2-6 shows interesting results for $\theta \approx 2.1$ with around 0.82% test problems succeed with 100 simplex gradient estimates. The graph 2-7 shows the better solution found with $\theta = 2.1135$ and $d_{NM3(100)} = 82.375\%$ (5931 test problems). Therefore, $\theta = 2.1135$ will be used for the next numerical experiments.

2.3.4 Monotone vs Non-Monotone

Thanks to the definition of the test problems, the presentation of the comparison profile and the best parameter θ, we can present relevant results.

We will present here two graphs, the data profile and the performance profile for all the methods and with $\theta = 2.1135$. We have used $\tau = 10^{-5}$ to define the desired accuracy.

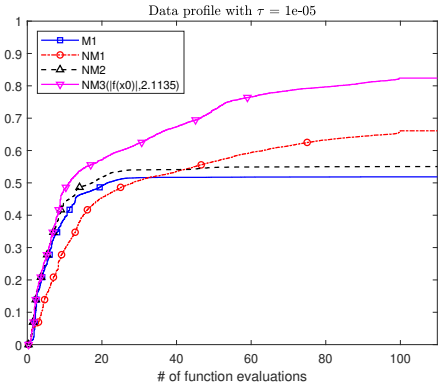


Figure 2-8: Data profile with $100(n_p + 1)$ simplex gradient estimates allowed

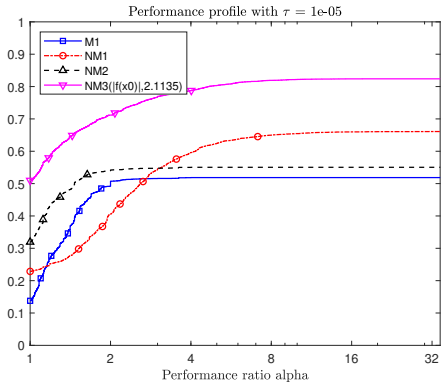


Figure 2-9: Performance profile with $100(n_p + 1)$ simplex gradient estimates allowed

The numerical results of our methods M1, NM1,NM2 and NM3 on the 7200 test

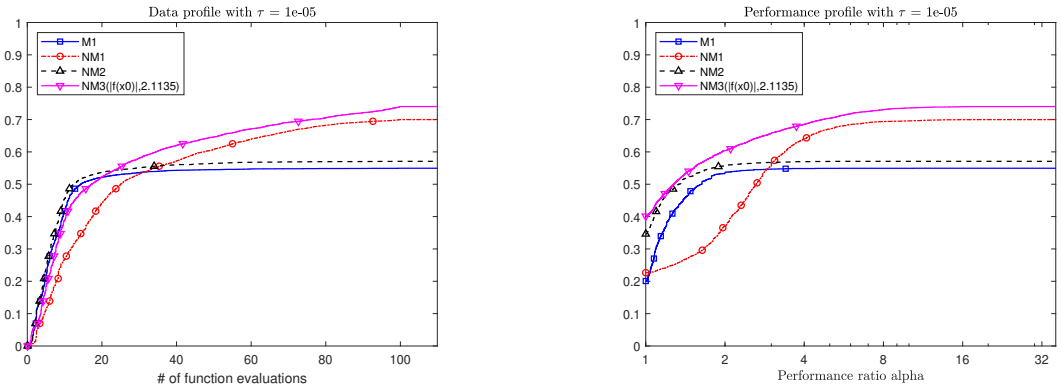
problems can be seen on figures 2-8 and 2-9. First, let's analyze the data profile 2-8, where we can clearly see that all non-monotone methods give better results than M1. In fact, M1 manages to solve 51.8% of the problems, while the results for NM1, NM2 and NM3 are 66%, 55% and 82.4% respectively.

We can also see that before 32 function evaluations, the NM1 algorithm is worse than any other, but the M1 and NM2 algorithms overtake it. The algorithm NM3 is better than the others from a distance, especially after 20 function evaluations.

We can now focus on the performance profile 2-9. The values $\rho_s(1)$ give the percentage of problems where the solver s was the best. Again, all non-monotone methods are better than M1 by this criterion. More precisely, M1 is best or equal on 10.1% of the test problems, while NM1, NM2 and NM3 are best or equal on 22.9%, 32.1% and 51% respectively. This sum doesn't equal one because different methods sometimes find the best solution with the same number of function evaluations. We can also see that NM3 is again the best.

The graphs above are made with the starting points generated with the general procedure. We can also investigate the performances of the methods with random starting points. We have generated these points with *rng(1,'twister')*.

Figure 2-10: Data and performance profile with $100(n_p+1)$ simplex gradient estimates allowed with random initial points



In the figure 2-10, the results are pretty similar to the ones presented in 2-8 and 2-9. We still have that the non-monotone methods are better than M1 on the data and performance profiles. Moreover, NM3 is still the best non-monotone algorithm, even if it is less successful than before.

Another more intuitive way to compare these algorithms is to use the box plot. With these plots, we only have information about the best values found with less than 100 simplex gradient estimates. The plots are in the annex B.

All the plots show well the solution distribution, which problem is solved and which remains unsolved. When we can see better the results found for each problem, the differences between the algorithms are less clear. We can't see the number of iterations needed to find these values. This highlights the strengths of the performance and data profile graphs.

2.3.5 Comparison with Simulated Annealing (SA)

As on the preliminary numerical results, we can compare our methods with the SA algorithm. We will focus on the methods M1, NM3 and SA to improve the readability of the graphs. We have used box plots to compare them as we don't have access to the history of the function value of *simulannealbnd*. Again these graphs can be seen in the annex B.

We present an overview of the results by considering the number of test problems each method solves. A method on a test problem is considered as successful if it satisfies the desired accuracy (2.1).

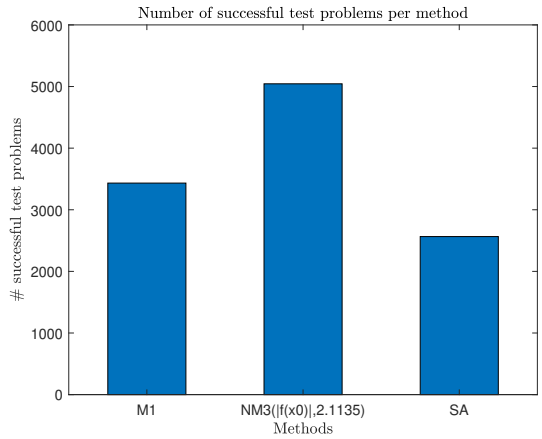


Figure 2-11: Comparison of M1, NM3 and SA in terms of number of test problems solved with 100 simplex gradient estimates

The bar chart 2-11 shows the competitiveness of local search against simulated annealing. With our choice of problems and the number of function evaluations allowed, even M1 succeeds in more test problems than SA. The non-monotone method NM3 keeps its interest by leading by far.

An unfair comparison:

We can say that even if we use the same number of function evaluations on the same test problems, comparing our methods and SA is unfair. Indeed, NM3 uses additional information than SA, as it is a first-order method, but the gradient computation is not considered. Moreover, the number of function evaluations can also lead to inequalities between the methods. NM3 will probably obtain better results than SA for a small number of function evaluations as a local search will go quickly into a local minimum. Conversely, SA will likely give better results for many function evaluations allowed as it will always find the solution when NM3 will always be stuck into a local minimum at the end.

A perfect comparison isn't possible, which is why we have to remember the key message of these comparisons. They show that considering the additional gradient information for a problem with limited function value allowed is interesting.

2.4 Summary of Conclusions

This part aimed to show the usefulness of the non-monotone algorithm over a monotone algorithm for several problems.

This objective is achieved in figures 2-8, 2-9 and 2-10. We have chosen a starting point set and analyzed the algorithm with data profiles, performance profiles and box plots. All these comparisons confirm the superiority of the non-monotone methods over M1 for the non-convex problems. Moreover, with a good choice of parameters, NM3 has shown impressive results compared to the others.

Finally, we have shown on the box plot 2-11 and the box plots in B that algorithms based on gradient descent are competitive with general algorithms for non-convex problems such as the Simulated Annealing algorithm.

Chapter 3

Better non-monotone method with best function value information

This chapter focuses on developing a new non-monotone algorithm for problems where the optimal value of the function is known. We propose a new algorithm to use this information more flexibly. A brief analysis of what has changed in the theory is done. Some numerical experiments are carried out to validate our choices.

3.1 More knowledge, better algorithms

In many situations, we can find more information by simply analyzing the function to be optimized. In our case, we will focus on problems where we can easily find the optimal function value. We know the best function value for all the problems presented in annex A, so we will also use them to analyze our results for this part.

First, let's explain qualitatively the expected behavior of this new algorithm. Thanks to the information about the value of the function, we can determine whether or not we are stuck in a local minimum. In fact, we can check both the accuracy of a found solution $f(x_k) - f^*$ and the gradient at that point $\nabla f(x_k)$, which tells us if we are at a critical point. These are valuable information when solving a non-convex global optimization problem.

Thanks to the works on non-monotone line search methods described in the previous chapters, we can allow our algorithm to accept bad iterates. NM1, NM2, and NM3 use this non-monotonic behavior to explore the function instead of falling directly into a local minimum. With this new information about the value of the function, we will use the non-monotonicity not to explore the function but to escape a local minimum. The difference between the two goals may be unclear, but we will clarify it soon.

3.2 Description of behavior

Now that the goal of escaping local minima is established, we will present a way to achieve this goal. We will design a new algorithm that retains the convergence results already obtained.

3.2.1 The local minima criterion

Before allowing non-monotone behavior, we must know whether the current iterate is close to a non-global local minimum. We already know that a necessary condition for a local optimizer is given by the theorem 1.3, $\nabla f(x) = 0$. Therefore, we will consider being in a local minimizer if the condition (3.1) is satisfied.

$$\frac{\|\nabla f(x_{k+1})\|}{\min\{f(x_0) - f^*, f(x_{k+1}) - f^*\}} \leq \delta \tag{3.1}$$

Thanks to the best function value information, we can distinguish non-global local minima from global minima. We consider that a non-global local minimum has the relative error given by (3.2).

$$\frac{f(x_{k+1}) - f^*}{f(x_0) - f^*} \geq \delta^2 \tag{3.2}$$

Remark 3.1. We have considered relative conditions in both cases not to be too influenced by function values. If considering a relative error is usual for the condition (3.2), the condition (3.1) is more unexpected.

However, it follows a desired behavior. In fact, if $f(x_{k+1}) - f^*$ is large but $\|\nabla f(x_{k+1})\|$ small, then we're likely to find a non-global minimum. Therefore, to speed up the algorithm, we don't allow too many iterations to find the non-global minimum with a precision of $\nabla f(x_{k+1}) \leq \delta$, but escape earlier thanks to the factor $f(x_{k+1}) - f^*$. This behavior is very interesting for some problems where finding an exact local minimum can be tedious. On the other hand, if $f(x_{k+1}) - f^*$ is small, both conditions will be hard to satisfy, and we will find an interesting local optimum. The minimum with $f(x_0) - f^*$ is used to avoid infinite value.

Remark 3.2. The choice of δ and δ^2 is not hazardous for (3.1) and (3.2). For a strongly convex function, we have the following relation,

$$f(x) - f^* \leq \frac{1}{2\mu}\|\nabla f(x)\|^2$$

Of course, we don't have any guarantee that the function will be strongly convex around the minimizer x. Nevertheless, this gives an expected relation between the gradient's norm and the function's error.

The main goal of this algorithm is only to consider a non-monotone behavior if both conditions (3.1) and (3.2) are satisfied, which means that we are stuck in a local minimum.

3.2.2 Direction of search

Several directions can be chosen to converge to a local minimum, as discussed in chapter 1. The one chosen for the moment is the quasi-Newton method BFGS. This choice converges to a local minimum with a superlinear convergence and is particularly adapted to find a minimum quickly.

A desired behavior of our new algorithm is to converge as fast as possible to a local minimum to check conditions (3.1) and (3.2). Thus, we can escape it and converge quickly to another one, and so on. Contrary to the previous non-monotone methods NM1, NM2, and NM3, we don't want to allow poor iterates when searching for local minima. Indeed, we prefer to take the most of the best function value information.

To achieve the performance of converging quickly to a local minimum, the quasi-Newton method BFGS is suitable. However, to escape a local minimum, a quasi-Newton step is not desired. Therefore, we will use $d_{k+1} = -\nabla f(x_{k+1})$, a gradient search, to escape non-global minima when conditions (3.1) and (3.2) are satisfied.

3.2.3 The step size

To escape a non-global minimum, we will use a special step size to escape local minima. The one chosen is the following,

$$\alpha_{k+1} = \begin{cases} \frac{R}{\|\nabla f(x_{k+1})\|} & \text{if (3.1) and (3.2) are satisfied} \\ \alpha_k \beta^{l_k - 1} & \text{else.} \end{cases}$$

This step size uses the usual definition to find a local minimum. When the conditions are satisfied, we use a new step size to escape. This new one has a parameter R at the numerator. The higher the value R, the easier it will be to escape local minima. However, taking a too important value for R may lead to irrelevant results. At the denominator, we have used the norm of the gradient. This is important as close to a local minimum; the update,

$$x_{k+2} = x_{k+1} + \alpha_{k+1} \beta^l d_{k+1}$$

may be negligible as $\|\nabla d_{k+1}\| \leq \min\{f(x_0) - f^*, f(x_{k+1}) - f^*\}\delta$ by condition (3.1). Finally, we don't consider α_k in the definition of α_{k+1} as it is not useful to escape the non-global minima.

3.2.4 The non-monotone term

Thanks to the new step size α_{k+1}, we can try to find an iterate outside the non-global minimum if conditions (3.1) and (3.2) are satisfied. However, the modified Armijo condition still has to accept this new iterate. This can be made more accessible by using the non-monotone term v_k.

After some numerical experiments, we have chosen,

$$v_{k+1} = \begin{cases} \sigma \min\{f(x_0) - f^*, f(x_{k+1}) - f^*\} \left(\frac{1}{1+k}\right)^{\phi} & \text{if (3.1) and (3.2) are satisfied} \\ 0 & \text{else.} \end{cases}$$

When escaping local minima, the goal is to have v_{k+1} big enough to satisfy the Armijo condition for a l,

$$f(x_{k+1} + \alpha_{k+1}\beta^l d_{k+1}) \leq f(x_{k+1}) + \rho\alpha_{k+1}\beta^l \nabla f(x_{k+1})^T d_{k+1} + v_{k+1}$$

even if $f(x_{k+1} + \alpha_{k+1}\beta^l d_{k+1}) \gg f(x_{k+1})$.

The parameter σ is given to the algorithm, allowing flexibility for the desired behavior. The term $\min\{f(x_0) - f^*, f(x_{k+1}) - f^*\}$ establishes the link between the current function value and the best function value. If the error $f(x_{k+1}) - f^*$ is huge, we want v_{k+1} also big. On the contrary, if the error $f(x_{k+1}) - f^*$ is small, we already get an interesting solution and want to reduce the non-monotone behavior. The part $f(x_0) - f^*$ is a constant which ensures that the term will always be finite. Finally, $\left(\frac{1}{1+k}\right)^{\phi}$ reduces the value of v_{k+1} as the iterations progress. The coefficient ϕ expresses how fast this decrease will be, with it the possibility of escaping known non-global local optima.

As we have already said, we aim to converge quickly to local minima. This is why we set $v_{k+1} = 0$ otherwise.

3.3 A non-monotone informed line search algorithm (NMI)

We can now define the algorithm NMI thanks to all the previous definitions.

Algorithm 2 Non-monotone informed descent algorithm

1: **Step 0** Given $x_0 \in \mathbb{R}^n$, $\alpha_0 > 0$, $\sigma \geq 0$, and $\beta, \rho, \delta \in (0,1)$, set $k := 0$, $v_0 = 0$ and $B_0 = I_{n,n}$.

2: **Step 1** Compute the descent direction $d_k \in \mathbb{R}^n$ for x_k as $d_k = -B_k \nabla f(x_k)$.

3: **Step 2.1** Set $l := 0$.

4: **Step 2.2** If

$$f(x_k + \alpha_k \beta^l d_k) \leq f(x_k) + \rho \alpha_k \beta^l \nabla f(x_k)^T d_k + v_k \tag{3.3}$$

set $l_k = l$ and go to Step 3.1. Otherwise, set $l := l + 1$ and repeat Step 2.2.

5: **Step 3.1** Set $x_{k+1} = x_k + \alpha_k \beta^{l_k} d_k$

6: **Step 3.2** Set $c = true$ if

$$\frac{f(x_{k+1}) - f^*}{f(x_0) - f^*} \geq \delta^2 \quad \text{and} \quad \frac{\|\nabla f(x_{k+1})\|}{\min\{f(x_0) - f^*, f(x_{k+1}) - f^*\}} \leq \delta$$

and $c = false$ otherwise.

7: **Step 3.3** Set,

$$v_{k+1} = \begin{cases} \sigma \min\{f(x_0) - f^*, f(x_{k+1}) - f^*\} \left(\frac{1}{1+k}\right)^{\phi} & \text{if } c = true \\ 0 & \text{else.} \end{cases}$$

$$\alpha_{k+1} = \begin{cases} \frac{R}{\|\nabla f(x_{k+1})\|} & \text{if } c = true \\ \alpha_k \beta^{l_k - 1} & \text{else.} \end{cases}$$

8: **Step 3.4**
Set $s_k = x_{k+1} - x_k$, $y_k = \nabla f(x_{k+1}) - \nabla f(x_k)$ and,

$$B_{k+1} = \begin{cases} I & \text{if } c = true \\ (I - \frac{s_k y_k^T}{s_k^T y_k}) B_k (I - \frac{y_k s_k^T}{s_k^T y_k}) + \frac{s_k s_k^T}{s_k^T y_k} & \text{else if } s_k^T y_k > 0 \\ B_k & \text{else} \end{cases}$$

9: **Step 4** Set $k = k + 1$ and go to Step 1

We have used the following values for the NMI method. We set $\sigma = \frac{M}{\delta^2 (f(x_0) - f^*)}$ with $M = 100$. This choice can balance the effect of the term,

$$\min\{f(x_0) - f^*, f(x_{k+1}) - f^*\}$$

when $f(x_{k+1}) - f^*$ is small to keep the non-monotonic behavior. Indeed, if we have $f(x_{k+1}) - f^* \approx \delta^2 (f(x_0) - f^*)$, this will cancel with the denominator of σ. Moreover, we have set $\phi = 1.01$. The value of ϕ affects the number of iterations needed to have $v_k \approx 0$ and thus a monotone behavior. Setting ϕ close to 1 allows more chances to find a better solution.

3.4 Worst case complexity analysis

We have shown in chapter 1 the worst-case complexity of Algorithm 1. We can also establish a worst-case complexity analysis for the new Algorithm 2.

First of all, like done in chapter 1, we will bound from below the step size α_k with lemma 3.1 slightly different to [2, Lemma 2].

Lemma 3.1. *Suppose that* ***A1*** *and* ***A3*** *hold. Then, for all k we have,*

$$\alpha_k \geq \min\left\{\alpha_0, \frac{R}{f(x_0)-f^*}, \frac{2(1-\rho)c_1}{Lc_2^2}\right\} \equiv \bar{\alpha}$$

Proof. We will prove the lemma by induction.

Let consider $k = 0$: The lemma is respected as the inequality $\alpha_0 \geq \min\{\alpha_0, ...\}$ is always true.

Suppose it is true for k and try to prove it for $k+1$. We have two different case, let first consider $\alpha_{k+1} = \frac{R}{\|\nabla f(x_{k+1})\|}$

$$\alpha_{k+1} = \frac{R}{\|\nabla f(x_{k+1})\|} \geq \frac{R}{\delta \min\{f(x_0)-f^*, f(x_{k+1})-f^*\}} \geq \frac{R}{f(x_0)-f^*} \geq \bar{\alpha}$$

where we used that $\delta \in (0,1)$.

We consider now the case $\alpha_{k+1} = \frac{\alpha_k}{\beta}$. This is the relation with $l_k = 0$,

$$\alpha_{k+1} = \frac{\alpha_k}{\beta} \geq \alpha_k \geq \bar{\alpha}$$

When the last inequality uses the recursion on k. We will now consider $l_k \geq 1$. We begin with the equation (3.3) in Algorithm 2 where we introduce $\alpha_{k+1} = \alpha_k \beta^{l_k - 1}$ and rearrange the terms.

$$f(x_k + \alpha_{k+1} d_k) - f(x_k) \geq \rho \alpha_{k+1} \nabla f(x_k)^T d_k + v_k$$

We can use the following well-known inequality,

$$|f(y) - f(x) - \nabla f(x)^T (y-x)| \leq \frac{L}{2}\|y-x\|^2$$

It is used on the left part with $y = x_k + \alpha_{k+1} d_k$ and $x = x_k$ to obtain,

$$\begin{aligned} f(x_k + \alpha_{k+1} d_k) - f(x_k) &\leq \nabla f(x_k)^T (x_k + \alpha_{k+1} d_k - x_k) + \frac{L}{2}\|\alpha_{k+1} d_k\|^2 \\ &= \alpha_{k+1} \nabla f(x_k)^T d_k + \frac{L\alpha_{k+1}^2}{2}\|d_k\|^2 \end{aligned}$$

When we join the inequalities together, we find,

$$\alpha_{k+1}\nabla f(x_k)^T d_k + \frac{L\alpha_{k+1}^2}{2}\|d_k\|^2 \geq \rho\alpha_{k+1}\nabla f(x_k)^T d_k + v_k$$

We can now rearrange the terms in the equation above,

$$\begin{aligned}
\alpha_{k+1} &\geq \frac{2}{L\|d_k\|^2}\left((\rho-1)\nabla f(x_k)^T d_k + \frac{v_k}{\alpha_{k+1}}\right) \\
&\geq \frac{2(1-\rho)}{L}\frac{-\nabla f(x_k)^T d_k}{\|d_k\|^2} + \frac{2v_k}{L\|d_k\|^2\alpha_{k+1}} \\
&\geq \frac{2(1-\rho)}{L}\frac{c_1\|\nabla f(x_k)\|^2}{c_2^2\|\nabla f(x_k)\|^2} && \text{We have used } \mathbf{A2} \text{ two times} \\
&= \frac{2(1-p)c_1}{Lc_2^2} && \text{By simplifying the term}
\end{aligned}$$

We find the third term of the lemma. ■

Thanks to lemma 3.1, we are now able to prove a worst-case complexity for the Algorithm 2 with the theorem 3.2 inspired from [2, Theorem 2]

Theorem 3.2. *Suppose that* ***A1-A4*** *hold and let the sequence* $\{x_k\}_{k\geq 0}$ *be generated by Algorithm 2. If,*

$$\sum_{k=0}^{T-1} v_k < +\infty$$

and,

$$T \geq \frac{\sum_{k=0}^{T-1} v_k + f(x_0) - f^*}{\kappa_c \epsilon^2} \tag{3.4}$$

with $\kappa_{\max} = \rho c_1 \bar{\alpha}\beta^{\max\{l_{\max},1\}}$ *then,*

$$\min_{k=0,\ldots,T-1}\|\nabla f(x_k)\| \leq \epsilon$$

Proof. Let assume by contradiction that $\min_{k=0,\ldots,T-1}\|\nabla f(x_k)\| > \epsilon$. We begin with equation 3.3 of Algorithm 2.

$$f(x_k + \alpha_k\beta^{l_k}d_k) \leq f(x_k) + \rho\alpha_k\beta^{l_k}\nabla f(x_k)^T d_k + v_k$$

Which leads to,

$$\begin{aligned}
v_k + f(x_k) - f(x_{k+1}) &\geq \rho\alpha_k\beta^{l_k} f(x_k, d_k)^T d_k \\
&\geq \rho\alpha_k\beta^{l_k} c_1\|\nabla f(x_k)\|^2
\end{aligned}$$

Now, we will consider the following sum,

$$\begin{aligned}\sum_{k=0}^{T-1} \rho\alpha_k\beta^{l_k}c_1\|\nabla f(x_k)\|^2 &\leq \sum_{k=0}^{T-1} v_k + f(x_k) - f(x_{k+1}) \\ &= f(x_0) - f(x_T) + \sum_{k=0}^{T-1} v_k \\ &\leq f(x_0) - f^* + \sum_{k=0}^{T-1} v_k\end{aligned}$$

We prove here that $\sum_{k=0}^{T-1} v_k \leq +\infty$, i.e. the series $\{v_k\}_{k\geq 0}$ is summable.

$$\sum_{k=0}^{T-1} v_k = \sum_{k=0}^{T-1} \sigma \min\{f(x_0) - f^*, f(x_k) - f^*\} \left(\frac{1}{1+k}\right)^{\phi} \leq \sum_{k=0}^{T-1} \sigma(f(x_0) - f^*) \left(\frac{1}{1+k}\right)^{\phi}$$

Where the last term is summable if $\phi > 1$.

By using the fact that $\|\nabla f(x_k)\| > \epsilon \ \forall \ k \in \{0, T-1\}$,

$$\sum_{k=0}^{T-1} \rho\alpha_k\beta^{l_k}c_1\epsilon^2 < f(x_0) - f^* + \sum_{k=0}^{T-1} v_k$$

We can simplify the left part like this,

$$\sum_{k=0}^{T-1} \rho\alpha_k\beta^{l_k}c_1\epsilon^2 = \rho c_1\epsilon^2 \left(\sum_{k=k_1}^{T-1} \alpha_k\beta^{l_k} + \sum_{k=k_2}^{T-1} \alpha_k\beta^{l_k}\right)$$

where we define $k_1 : k_1 + 1$ respects conditions (3.1) and (3.2) and $k_2 : k_2 + 1$ doesn't. Both sums can be simplified,

$$\sum_{k=k_1}^{T-1} \alpha_k\beta^{l_k} \geq \sum_{k=k_1}^{T-1} \bar{\alpha}\beta^{l_k} \geq \sum_{k=k_1}^{T-1} \bar{\alpha}\beta^{l_{\max}}$$

By taking $l_{\max} = \max_{k=k_1} l_k$.

$$\sum_{k=k_2}^{T-1} \alpha_k\beta^{l_k} \geq \sum_{k=k_2}^{T-1} \bar{\alpha}\beta$$

Therefore, we have,

$$\rho c_1\epsilon^2 \left(\sum_{k=k_1}^{T-1} \bar{\alpha}\beta^{l_{\max}} + \sum_{k=k_2}^{T-1} \bar{\alpha}\beta\right) \geq \rho c_1\epsilon^2 T\bar{\alpha}\beta^{\max\{l_{\max},1\}} \geq \kappa_{\max} T\epsilon^2$$

with,

$$\kappa_{\max} = \rho c_1\bar{\alpha}\beta^{\max\{l_{\max},1\}}$$

Finally, by rearranging the terms, we get the final inequality,

$$T < \frac{f(x_0) - f^* + \sum_{k=0}^{T-1} v_k}{\kappa_{\max} \epsilon^2}$$

■

We have proved here that the algorithm NMI has a worst-case complexity in $\mathcal{O}(\epsilon^{-2})$.

3.5 Numerical results

Now that we have described the new non-monotonically informed method NMI, we can compare it with the others on the 7200 test problems. For this comparison we have used $\delta = 10^{-3}$, $R = 1$ and $M = 100$.

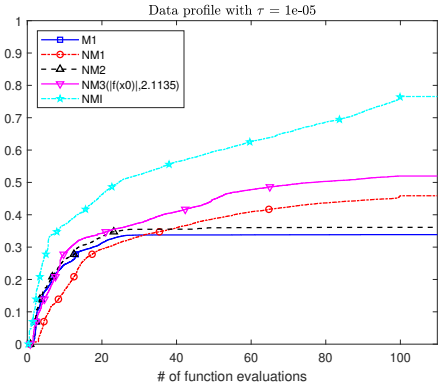


Figure 3-1: Data profile comparison with NMI

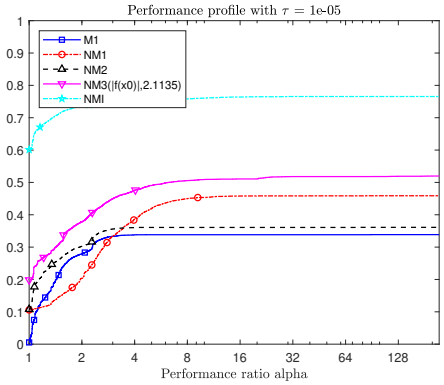


Figure 3-2: Performance profile comparison with NMI

The graphs 3-1 and 3-2 clearly show that using the informed algorithm gives better results than our previous non-monotone algorithm.

In more detail, let us first analyze the data profile 3-1. At 5 function evaluations, all algorithms are neck-and-neck. With only a few iterations, the non-monotone part, which is the difference between the algorithms, may not have been spoken yet. After 5 function evaluations, the non-monotone informed algorithm (NMI) takes the lead. It finds solutions faster and more than the other algorithms, solving about 76% of the problems when the others are stuck below 52%. The performance profile supports this analysis. We can see that NMI is the best on 60% of the problems and is thriving on 16% other problems. This shows that using all non-monotone ideas to exploit the value of the function doesn't slow down the algorithm and gives it more power to find interesting solutions.

Chapter 4

The Euclidean Distance Geometry Problem

This chapter will look at the famous Molecular Distance Geometry Problem (MDGP). With this new challenge, we aim to test the new algorithm NMI developed in chapter 3. We will begin with benchmark problems and then use real proteins available on the Protein Data Bank [3]. The results will be compared to some previous methods, namely M1 and NM3. Finally, a quality indicator of the solution found will be explained and used.

4.1 Molecular Distance Geometry Problem (MDGP)

The problem of interest we will study in this Master's thesis is the Euclidean Distance Geometry Problem (DGP). It was defined in the 20th century by Menger [23], Blumenthal [24] and others. We will consider here a version of this problem called the Molecular Distance Geometry Problem (MDGP), which is the same problem with the added context of molecular structure.

The context of this problem is the following, finding the spatial structures of a molecule is a subject of interest in the field of biochemical research. Stereochemistry [25] is a discipline that studies the structure of molecules to understand their impact on their behavior. This is particularly useful in the pharmaceutical field, where it has dramatically helped. We can cite the case of thalidomide, the drug used to treat morning sickness in women, which turned out to be teratogenic, i.e. it caused genetic damage to the baby. In addition, the structure of the molecules gives the bond angle and length, which allow interesting measurements to be made. They use nuclear magnetic resonance (NMR) spectroscopy to obtain this structure. This technique allows the distances between close pairs of atoms in the molecule to be determined with some accuracy.

In our context, we can define $K = 3$ as the dimension of the atomic position. We have d_{uv} the distance between the atoms u and v and x_u, x_v their respective positions in $\mathbb{R}^3$. The set E contains all pairs of atoms u, v for which the distance d_{uv} is known.

The mathematical formulation of the MDGP can be written like this as written in [26].

$$\min_{x \in \mathbb{R}^K} f(x) := \sum_{\{u,v\} \in E} (\|x_u - x_v\|^2 - d_{uv}^2)^2 \tag{4.1}$$

The challenge of this method, as described by the minimization problem (4.1), is to

find the set of atomic positions that match the known distances. We can note that the objective function of the problem (4.1) is an error function with minimum value $f(x) = 0$. This property will allow us to use the new non-monotone method informed NMI designed in chapter 3.

Before implementing a solution to this problem, let us try to understand its characteristics. The first idea is to understand the critical points of the function described in (4.1). We choose the notation $[x_k]_i$ to refer to the i^{th} component of the atom k.

$$\nabla f([x_k]_i) = \sum_{(k,v)\in E} 4([x_k]_i - [x_v]_i)(\|x_k - x_v\|^2 - d_{k,v}^2)$$

Each critical point x must satisfy $\nabla f([x_k]_i) = 0 \ \forall\, k, i$ as established by the theorem 1.3.

We can already highlight some critical points thanks to the gradient's shape. We know that there is an arbitrary set of positions x which satisfy,

$$[x_k]_i - [x_v]_i = 0 \quad \text{or} \quad \|x_k - x_v\|^2 - d_{k,v}^2$$

for each gradient's components. All of them are critical points.

The first possibility is to have $\|x_k - x_v\| = d_{k,v} \ \forall\, (k, v) \in E$. This solution x is a global minimum since it leads to $f(x) = 0$. This solution is not unique since any translation or rotation of the set of atoms will lead to the same result.

A second possibility is that $[x_k]_i - [x_v]_i = 0 \ \forall\, (k, v) \in E$ and i. This solution can be written on the form,

$$x_k = (z_1, z_2, z_3) \ \forall\, k$$

with z_1, z_2, z_3 constants.

We can analyze the hessian to determine the type of critical points as described in the theorem 1.6.

$$\begin{aligned}
\frac{\partial^2 f(x)}{\partial^2 [x_k]_i} &= \sum_{(k,v)\in E} 8([x_k]_i - [x_v]_i)^2 + 4(\|x_k - x_v\|^2 - d_{k,v}^2) \\
\frac{\partial^2 f(x)}{\partial [x_k]_i \partial [x_k]_j} &= \sum_{(k,v)\in E} 8([x_k]_j - [x_v]_j)([x_k]_i - [x_v]_i) \\
\frac{\partial^2 f(x)}{\partial [x_k]_i \partial [x_v]_i} &= -8([x_v]_i - [x_k]_i)^2 - 4(\|x_k - x_v\|^2 - d_{k,v}^2) \\
\frac{\partial^2 f(x)}{\partial [x_k]_i \partial [x_v]_j} &= -8([x_v]_j - [x_k]_j)([x_v]_i - [x_k]_i)
\end{aligned}$$

Therefore, if $[x_k]_i - [x_v]_i = 0 \ \forall\, (k, v) \in E$ and i then the hessian has the following form.

$$H = \begin{pmatrix} -\sum_{(1,v)\in E} d_{1,v}^2 & 0 & 0 & d_{1,2}^2 & \dots & d_{1,n}^2 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ d_{1,n}^2 & \dots & d_{n,n-1}^2 & 0 & 0 & -\sum_{(1,v)\in E} d_{n,v}^2 \end{pmatrix}$$

This hessian is called a diagonally dominant matrix as $|h_{i,i}| \geq \sum_{j\neq i} a_{i,j}$. If all diagonal elements are strictly negative, the matrix is negative semidefinite. This means this critical corresponds to a local maximum or a saddle point. We can easily see that moving one atom away from the others can only decrease the error function since the available distances can only be positive. Therefore this solution is a local maximum.

Even if this local maximum is greater than or equal to the other local maxima, it is not a global maximum because the function $f(x)$ can take infinite value for terrible solutions, e.g. with a norm of $\|x_u - x_v\| \longrightarrow +\infty$.

There are numerous other ways of satisfying the conditions (4.1), i.e. all combinations of the two possibilities above. We can imagine a set of atoms where $x_u = x_v \ \forall\, u, v \neq k$ and $\|x_k - x_u\| = d_{k,u}^2$ for a given pair (k, u). The hessian is no longer diagonally dominant, and numerical results show positive and negative eigenvalues. These points are, therefore, saddle points.

Finally, there exist other possibilities to obtain $\nabla f(x) = 0$ than by satisfying conditions (4.1). Elements of the sum can cancel each other to get a zero gradient. A nonlinear system has to be solved to find these solutions. Unfortunately, there are no other solutions for small systems with six variables (2 atoms in 3 dimensions or 3 atoms in 2 dimensions), and larger systems can't be solved with all the solutions. However, we can easily imagine many local minima hidden in this system. Indeed, the problem MDGP is known to be particularly difficult with many local minima.

We conclude this presentation of the MDGP by considering three different data sets.

- The distances $d_{u,v}$ are available $\forall\, (u, v)$.
- The distances $d_{u,v}$ are only available $\forall\, (u, v) \in E$ with E a subset of all the possible distances.
- The distances $d_{u,v}$ are known up to a certain precision,

  $$l_{u,v} \leq d_{u,v} \leq L_{u,v}$$

  with $l_{u,v}$ and $L_{u,v}$ known.

In this work, only the first two data sets will be analyzed. The third data set can be chosen as an advanced topic to explore. In this case, a different objective function should be used, as proposed by Souza et al. [27].

$$f(x) = \sum_{(u,v)\in E} p_{u,v}(x)(x_u - x_v)$$

with,

$$p_{u,v}(x) = \max\left\{\frac{l_{u,v}^2 - \|x\|^2}{l_{u,v}^2}, \frac{\|x\|^2 - L_{u,v}^2}{L_{u,v}^2}, 0\right\}$$

4.2 Numerical experiments for benchmark problems

In our first numerical experiments, we will test a subset of the methods discussed in chapter 2 and chapter 3 on a set of artificial problems proposed by Moré and Wu [28]. These problems have been reused to compare other algorithms in papers such as Haw-ren F., Dianne P. and O'Leary [29] or Liberti et al. [26].

These problems are defined for $m = s^3$ atoms, where the position of each atom is given by

$$(u_1, u_2, u_3) : u_1, u_2, u_3 = 0, ..., s-1$$

We can assign a unique index to each atom using the following mapping,

$$u = 1 + u_1 + u_2 s + u_3 s^2$$

Now that the set of atoms is properly defined, we can calculate the Euclidean distance between them. To mimic a NMR experiment, some distances must be forgotten. In our case, all distances greater than 2 have been removed to form the subset of distances E.

$$E = \{(u, v) : \|x_u - x_v\| \leq 2\}$$

where x_u are the coordinates of atom u.

The initial set of points is very important for this problem. In fact, during our analysis of the problem, we noticed that starting with the same value for the components of the atoms leads to a local maximum from which it is impossible to escape. Therefore, the generic grid used in the previous chapter doesn't work well in practice for this problem. The starting points suggested by Moré and Wu [22] are as follows

$$\{x_u \in \mathbb{R}^3 : 0 \leq [x_u]_i \leq s-1\} \tag{4.2}$$

where each component i is randomly chosen in $[0, s-1]$. The randomness used for the following experiments is set by *rng(1,'twister')*.

For Moré and Wu [28], a problem is considered solved if

$$||x_u - x_v| - d_{u,v}| < 0.01 \ \forall (u, v) \in E$$

To focus on the value of the function, we consider $f(x) \leq C\epsilon$, where $\epsilon = 10^{(-5)}$ and $C = |E|$ the cardinality of the subset of the available distance E. The goal is to have approximately

$$((x_u - x_v)^2 - d_{u,v}^2)^2 \leq \epsilon \ \forall (u, v) \in E$$

4.2.1 The different results

The methods used in the following numerical comparisons are M1, NM3 and NMI. We recall their definition,

- M1 is a monotone method following the Algorithm 1 with $v_k = 0 \ \forall k$.
- NM3 is a non-monotone method following the Algorithm 1 with $v_{k,l}$ defined as,

$$v_{k,l} = \sigma \exp\left(-\frac{\max(\theta, f(x_k + \alpha_k \beta^l d_k) - f(x_k))}{\tau_k}\right)$$

  where, $\tau_k = \frac{1}{\ln(k+1)}$ We use $\sigma = |f(x_0)|$ and $\theta = 2.1135$ for the rest of this chapter.
- NMI is a non-monotone method informed by the algorithm 2 and presented in chapter 3. We use the parameters $M = 10^6$ and $R = 10$. These parameters were found after numerical experiments.

We ran the comparison for 30 different random starting points defined by the rule (4.2), with $100(n_p + 1)$ iterations allowed. When the stopping criterion $f(x) \leq C\epsilon$ is reached, the experiments are stopped. The results with $s = 3$ and $s = 4$ are as follows,

Figure 4-1: Benchmark problem with $s = 3$ and $100(n_p + 1)$ iterations allowed

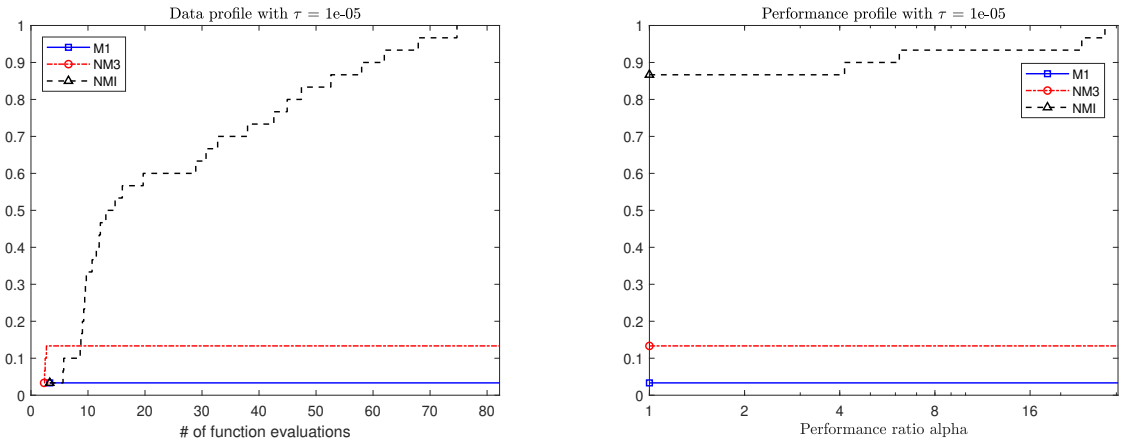
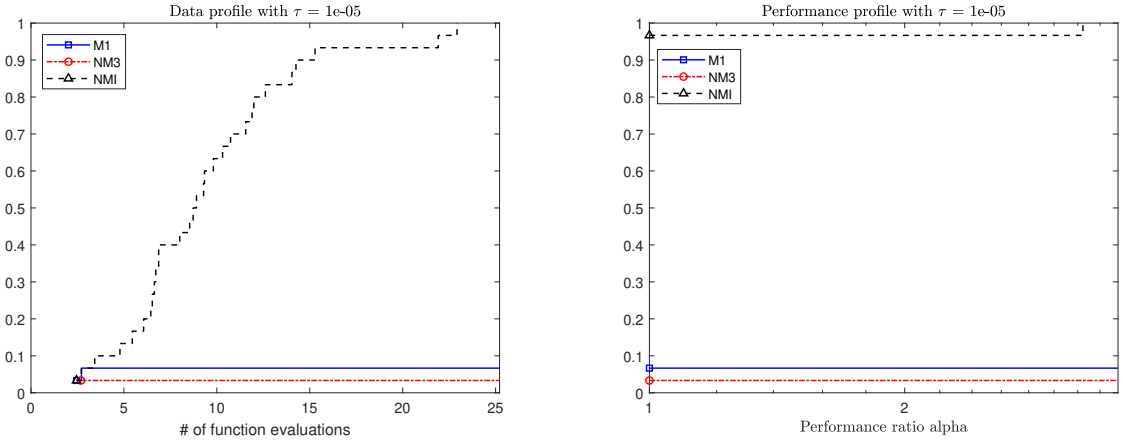


Figure 4-2: Benchmark problem with $s = 4$ and $100(n_p + 1)$ iterations allowed



The results of 4-1 and 4-2 clearly show that NMI is much better than the others. In the table 4.1, we have noted the number of successful test problems per method.

Table 4.1 Comparison table for the benchmark problem

s	3	4
|E|	185 (52.71%)	564 (27.98%)
M1 #Sucess	1	2
NM3 #Sucess	0	1
NMI # Sucess	24	29

Again, the results are clear. Table 4.1 shows that only the informed method finds acceptable solutions. Surprisingly, the methods perform better with $s = 4$ and fewer known distances. However, we can't interpret this result as general. The NM3 algorithm doesn't seem good enough to find an acceptable solution. Its non-monotone behavior may be adaptable to give better results.

4.3 Numerical results with proteins

Now that we have verified the performance of our algorithms on a benchmark problem, it is time to consider new challenges. The main goal in developing this informed algorithm was to test it on those real-world problems, the molecules.

The Protein Data Bank [3] shows all the data used for the following tests. One of the main difficulties is the size of the problem. Molecules and proteins can have anywhere from 100 to over 300000 atoms. To keep computation times reasonable, we will focus on the smallest proteins.

We have used this set of parameters to obtain the following results. We chose $R = 50$ after some numerical experiments. Moreover, using real proteins with many atoms dramatically increases the possible function values obtained. We also increased the

value of the constant M to $M = 10^{10}$. As in the benchmark problem, we have to use a value of M large enough to escape the local minima.

The starting points for the following numerical results have been defined by,

$$\{x_u \in \mathbb{R}^3 : l \leq [x_u]_j \leq L\} \tag{4.3}$$

where each component is randomly chosen in $[l, L]$, where l and L are the smallest and largest component values of the solution given by the Protein Data Bank [3].

4.3.1 The solutions for some proteins

The first protein test is one of the smallest; it is called *1a02* and has 147 atoms. The file provided by the Protein Data Bank [3] gives the position of the atoms in 3 dimensions. To simulate NMR data, we will only consider distances between atoms of less than 6 Angstroms, as proposed in Amaral et al. [30]. This resulted in $|E| = 3425$, representing 3.13% of the total distances.

With the first random points defined by the rule (4.3) and *rng(1,'twister')*, NMI succeeds in finding a point x_k such that $f(x_k) \leq C\epsilon$ with $\epsilon = 10^{-5}$ and $100(n_p + 1)$ iterations allowed.

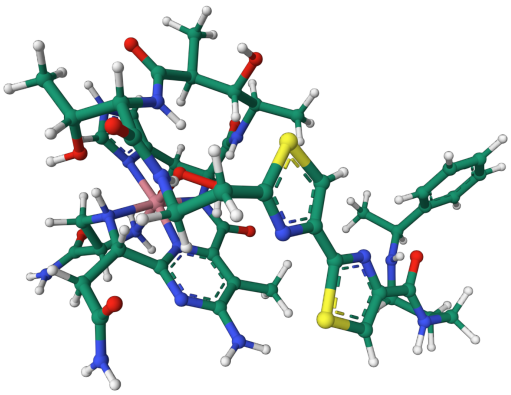


Figure 4-3: Original protein *1a02*

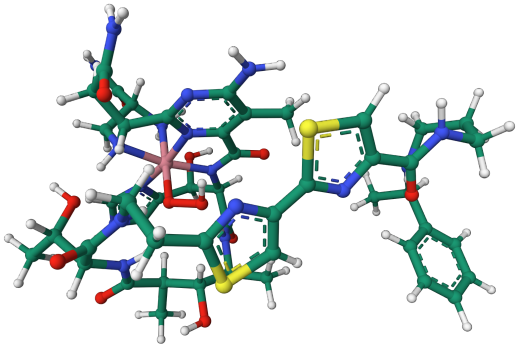


Figure 4-4: Our solution for protein *1a02*

The two solutions 4-3 and 4-4 are very similar; the one found by our algorithm has a symmetry rotation with the original.

The second protein we are interested in is *1ptq* and has 402 atoms. This problem is more significant with $|E| = 7088$, corresponding to 11.43% of the total number of distances. Again, our algorithm succeeds in finding a point x_k such that $f(x_k) \leq C\epsilon$ with $\epsilon = 10^{-5}$ and $100(n_p + 1)$ iterations allowed.

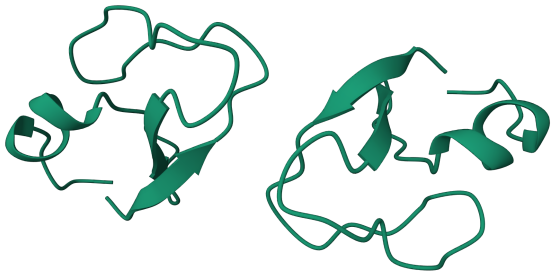


Figure 4-5: Original protein *1ptq*

Figure 4-6: Our solution for protein *1ptq*

Again, the two molecules appear to be different but are similar. The molecules are made up of two separate parts. Our solution has brought them closer together and rotated one of them.

4.3.2 Numerical comparison between the methods M1, NM3, NMI, SA and SA informed

The NMI method succeeds in finding interesting solutions as shown above for real proteins. Let's examine this performance with other methods. We will compare it with the M1 and NM3 methods as for the benchmark problem, but also with the Simulated Annealing (SA) method. We have differentiated the methods SA and SA as we give the parameter *Objective Limit* $= 0$ for the second one. We will consider protein *1ao2* for the following test. The methods are stopped when we have found a sufficiently good solution described by $|f(x_k) - f(x^*)| \leq C\epsilon$, where $f(x^*) = 0$ and $C\epsilon = 3.42510^{-2}$. The tests are run for ten different starting points generated by *rng(1,'twister')*.

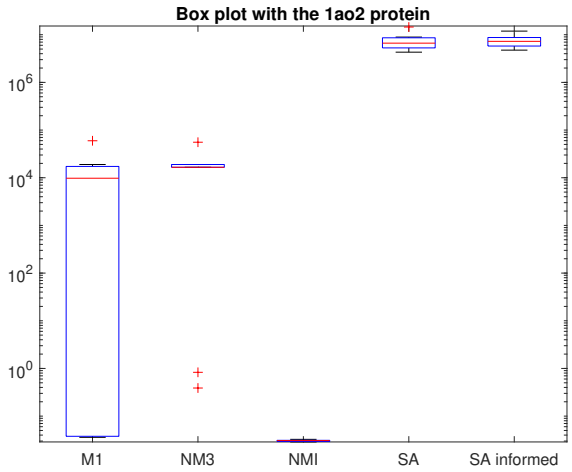


Figure 4-7: Comparison for *1ao2* protein and $100(n_p + 1)$ simplex gradients estimates allowed

On the graph 4-7 we can see that Simulated Annealing fails to find any interesting value, even when using the full $100(n_p + 1)$ simplex gradient estimates with $n_p = 441$ as *1ao2* has 147 atoms. The M1 method found a solution for 3 starting points, while NM3 didn't succeed in any of them. Finally, the NMI succeeds in all of them.

4.4 Quality of a solution

Measuring the quality of the solution found is not trivial. More than obtaining $f(x) = 0$ is required to say that we have found the solution since we only consider distances $\in E$. We have the initial solution to the problem, but we need to find a way to compare the two solutions. In fact, since the function $f(x)$ in (4.1) is invariant to rotation and translation, both solutions may be identical but seems different. Comparing the two solutions up to a specific rotation and translation is known as the Procrustes problem.

Amaral et al. [30] use the following method to solve this problem. We define X^* as the best solution found by our algorithm and X as the solution given by the Protein Data Bank [3].

The first step is to translate both solutions around the origin; this can be done by looking at the set of points defined as X^*J and XJ with $J = I - \frac{1}{n}ee^T$ and $e = (1, 1, ..., 1, 1)$.

Then we want to find the best rotation matrix to match both solutions. This can be done by solving the following optimization problem.

$$\begin{aligned} \min \ & \|RX^*J - XJ\|_F^2 \\ \text{s.t.} \ & RR^T = I \end{aligned}$$

The orthogonal matrix $Q \in \mathbb{R}^{3x3} = R^*$ has a closed-form solution. This is given by $Q = VU^T$ where $U\Sigma V^T$ is the SVD of the matrix $X^*J(XJ)^T$. So we can easily compare the two solutions X^*J and XJ, equivalent to comparing X^* and X.

There are several ways to define a measure of the error of X^*. The one considered by Amaral and al. [30] is a relative error that takes an interest in the maximum error of each atom.

$$E(X^*) = \max_{j=1,...,n} \left\{ E(x_j^*) \right\}$$

with,

$$E(x_j^*) = \frac{\|[QX^*J - XJ]_j\|_\infty}{\max\{1, \|[XJ]_j\|_\infty\}}$$

where the coefficient j denotes the column of the matrix.

4.4.1 Analysis of quality

We can examine the quality of the two solutions found for the *1ao2* and the *1ptq* proteins. We recall that these experiments were carried out with a set of distances smaller than 6 Angstroms and a stopping criterion $f(x_k) \leq C\epsilon$, where $\epsilon = 10^{-5}$ and $C = |E|$, the number of available distances.

Again, we have chosen an initial solution defined by the rule (4.3) with *rng(1,'twister')*. The solution the algorithm NMI returns is called X^*. For the protein *1ao2* we found $E(X^*) = 2.79e^{-4}$ and for the protein *1ptq* we found $E(X^*) = 1.48e^{-2}$. Both solutions give satisfactory results as the difference can't be seen with the naked eye on the graphs 4-8 and 4-9.

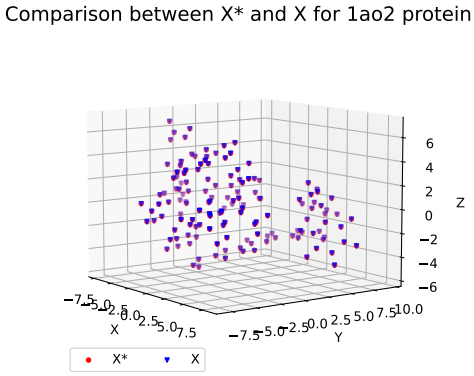


Figure 4-8: Quality of the solution for the protein *1ao2*

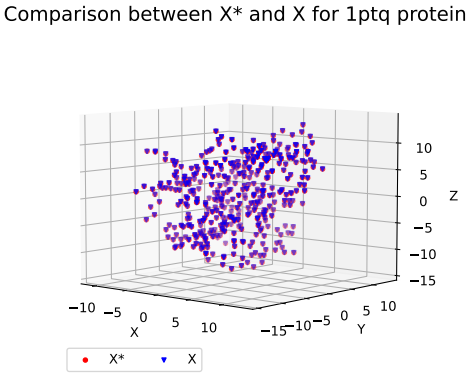


Figure 4-9: Quality of the solution for the protein *1ptq*

Conclusion

In this work, we presented an extensive numerical comparison between monotone and non-monotone line search methods. Specifically, we compared the monotone Armijo line search against the non-monotone variants proposed in [4, 5, 2] on a set of unconstrained optimization problems with many non-global local minimizers. The results show that the classical non-monotone method proposed by Grippo, Lucidi, and Lampariello [4], and the Metropolis-based method proposed by Grapiglia and Sachs [2] are significantly better than the monotone Armijo line search, exhibiting the ability to escape from non-global local minimizers. Inspired by these results, and building upon the general framework described in [2], we proposed a new non-monotone method for unconstrained global optimization problems in which the optimal function value f^* is known a priori. A notable example is the Molecular Distance Geometry Problem, which consists in finding the position of atoms knowing a subset of the distances between them. For problems of this type, the new method uses the functional residual $f(x_k) - f^*$ to control the level ν_k of nonmonotonicity allowed in the line search condition. If $\|\nabla f(x_k)\|$ is close to zero, but $f(x_k) - f^*$ is still large, this indicates that the iterates are converging to a non-global local minimizer. When this situation is detected, the new method uses $\nu_k > 0$, to allow the iterates to escape that basin of attraction and move towards a possibly better local minimizer; otherwise, it sets $\nu_k = 0$. We show that the proposed method takes at most $\mathcal{O}\left(\epsilon^{-2}\right)$ iterations to find and ϵ-approximate stationary point of the objective function. Moreover, numerical results clearly show the benefits of exploiting the knowledge of f^* in the algorithmic design. An interesting question for future research is whether there exists a class of nonconvex functions with many non-global local minimizers for which one can guarantee the global convergence of non-monotone line search methods to global minimizers.

Bibliography

[1] G. N. Grapiglia and E. W. Sachs, "On the worst-case evaluation complexity of non-monotone line search algorithms," *Computational Optimization and Applications*, vol. 68, no. 3, pp. 555–577, 2017.

[2] G. Grapiglia and E. Sachs, "A Generalized Worst-Case Complexity Analysis for Non-Monotone Line Searches," 11 2019.

[3] https://www.rcsb.org/, "Protein data bank." Last accessed 16 March 2023.

[4] L. Grippo, F. Lampariello, and S. Lucidi, "A Nonmonotone Line Search Technique for Newton's Method," *SIAM Journal on Numerical Analysis*, vol. 23, no. 4, pp. 707–716, 1986.

[5] H. Zhang and W. W. Hager, "A Nonmonotone Line Search Technique and Its Application to Unconstrained Optimization," *SIAM Journal on Optimization*, vol. 14, no. 4, pp. 1043–1056, 2004.

[6] A. Cauchy, "Méthode générale pour la résolution des systèmes d'équations simultanées," *C. R. Acad. Sci. Paris*, vol. 25, pp. 536–538, 1847.

[7] Y. Nesterov, "Introduction Lectures on Convex Programming," vol. 1, p. 211, July 1998.

[8] J. Raphson, "Analysis aequationum universalis," 1690.

[9] J. E. Dennis and J. J. Moré, "A characterization of superlinear convergence and its application to quasi-newton methods," *Mathematics of Computation*, vol. 28, no. 126, pp. 549–560, 1974.

[10] L. Armijo, "Minimization of functions having Lipschitz continuous first partial derivatives.," *Pacific Journal of Mathematics*, vol. 16, no. 1, pp. 1 – 3, 1966.

[11] P.-A. Absil, R. Mahony, and B. Andrews, "Convergence of the Iterates of Descent Methods for Analytic Cost Functions," *SIAM Journal on Optimization*, vol. 16, no. 2, pp. 531–547, 2005.

[12] A. Jentzen and A. Riekert, "A proof of convergence for stochastic gradient descent in the training of artificial neural networks with ReLU activation for constant

target functions," *Zeitschrift für angewandte Mathematik und Physik*, vol. 73, p. 188, 2022.

[13] J. Z. Y. Y. Lihe C., Zhengyi S. and Y. G., 2021. Last accessed 25 April 2023.

[14] W. Philip, "Convergence conditions for ascent methods *,"

[15] G. Zoutendijk, "Nonlinear Programming, Computational Methods," *Integer and Nonlinear Programming*, pp. 37–86, 1970.

[16] M. Ahookhosh, K. Amini, and S. Bahrami, "A class of nonmonotone Armijo-type line search method for unconstrained optimization," *Optimization 61 (4), 387-404*, vol. 61, 01 2012.

[17] A. M. . N. H. Amini, K., "An inexact line search approach using modified nonmonotone strategy for unconstrained optimization," *Numer Algor*, vol. 66, p. 49–78, 2014.

[18] M. Ahookhosh and S. Ghaderi, "On efficiency of nonmonotone Armijo-type line searches," *Applied Mathematical Modelling*, vol. 43, pp. 170–190, 2017.

[19] C. Cartis, P. Sampaio, and P. Toint, "Worst-case evaluation complexity of non-monotone gradient-related algorithms for unconstrained optimization," *Optimization*, vol. 64, pp. 1–13, 12 2014.

[20] E. Sachs and S. Sachs, "Nonmonotone line searches for optimization algorithms," *Control and Cybernetics*, vol. 40, pp. 1059–1075, 01 2011.

[21] M. Ali, C. Khompatraporn, and Z. Zabinsky, "A Numerical Evaluation of Several Stochastic Algorithms on Selected Continuous Global Optimization Test Problems," *Journal of Global Optimization*, vol. 31, pp. 635–672, 01 2005.

[22] J. Moré and S. Wild, "Benchmarking Derivative-Free Optimization Algorithms," *SIAM Journal on Optimization*, vol. 20, pp. 172–191, 01 2009.

[23] K. Menger, "Untersuchungen über allgemeine Metrik," *Mathematische Annalen*, vol. 100, pp. 75–163, 1928.

[24] L. Blumenthal, "Theory and Applications of Distance geometry," 1953.

[25] Wikipedia, "Stereochemistry."

[26] L. Liberti, C. Lavor, N. Maculan, and A. Mucherino, "Euclidean Distance Geometry and Applications," *SIAM Review*, vol. 56, 05 2012.

[27] L. C. M. A. M. A. Souza, M., "Solving the molecular distance geometry problem with inaccurate distance data," *BMC Bioinformatics*, vol. 14, pp. 1471–2105, 2013.

[28] J. J. Moré and Z. Wu, "Global Continuation for Distance Geometry Problems," *SIAM Journal on Optimization*, vol. 7, no. 3, pp. 814–836, 1997.

[29] H. ren Fang and D. P. O'Leary, "Euclidean distance matrix completion problems," *Optimization Methods and Software*, vol. 27, no. 4-5, pp. 695–717, 2012.

[30] V. S. Amaral, R. Andreani, E. G. Birgin, D. S. Marcondes, and J. M. Martínez, "On complexity and convergence of high-order coordinate descent algorithms for smooth nonconvex box-constrained minimization," 2020.

Appendix A

Selection of global tests problems

This appendix contains the descriptions of the problems used in the paper. All these problems have been selected because they are twice differentiable everywhere and have many local minima. The 20 problems are proposed and others by Ali, Khompatraporn and Zabinsky [21]. The problems considered are the following,

Table A.1 Subset of the problem set proposed by [21]

	Name	Dimension
1.	Bohachevsky 1 Problem (B1)	2
2.	Bohachevsky 2 Problem (B2)	2
3.	Cosine Mixture Problem (CM)	4
4.	Easom Problem (EP)	2
5.	Epistatic Michalewicz Problem (EM)	10
6.	Exponential Problem (EXP)	10
7.	Griewank Problem (GW)	2
8.	Levy and Montalvo 1 Problem (LM1)	3
9.	Levy and Montalvo 2 Problem (LM2)	10
10.	Modified Langerman Problem (ML)	10
11.	Neumaier 2 Problem (NF2)	4
12.	Neumaier 3 Problem (NF3)	10
13.	Price's Transistor Modelling Problem (PTM)	9
14.	Rastrigin Problem (RG)	10
15	Schaffer 1 Problem (SF1)	2
16.	Schaffer 2 Problem (SF2)	2
17.	Shekel's Foxholes Problem (FX)	10
18.	Shubert Problem (SBT)	2
19.	Sinusoidal Problem (SIN)	10
20.	Storn's Tchebychev Problem (ST)	9

A.1 Bohachevsky 1 Problem

The problem is

$$\min_x f(x) \ = x_1^2 + 2x_2^2 - 0.3\cos(3\pi x_1) - 0.4\cos(4\pi x_2) + 0.7$$

with,

$$\nabla f(x) = \begin{pmatrix} 2x_1 + 0.9\pi \sin(3\pi x_1) \\ 4x_2 + 1.6\pi \sin(4\pi x_2) \end{pmatrix}$$

with $x^* = (0, 0)$ as a global minimizer and $f(x^*) = 0$. The relevant hypercube is $-50 \leq x_1, x_2 \leq 50$.

A.2 Bohachevsky 2 Problem

The problem is

$$\min_x f(x) \ = x_1^2 + 2x_2^2 - 0.3\cos(3\pi x_1)\cos(4\pi x_2) + 0.3$$

with,

$$\nabla f(x) = \begin{pmatrix} 2x_1 + 0.9\pi \sin(3\pi x_1)\cos(4\pi x_2) \\ 4x_2 + 1.2\pi \sin(4\pi x_2)\cos(3\pi x_1) \end{pmatrix}$$

with $x^* = (0, 0)$ as a global minimizer and $f(x^*) = 0$. The relevant hypercube is $-50 \leq x_1, x_2 \leq 50$.

A.3 Cosine Mixture Problem

The problem is

$$\min_x f(x) \ = -0.1 \sum_{i=1}^{n} \cos(5\pi x_i) + \sum_{i=1}^{n} x_i^2$$

with,

$$\frac{\partial f}{\partial x_i} = 0.5\pi \sin(5\pi x_i) + 2x_i$$

The dimension considered is $n = 4$, with $x^* = (0, 0, 0, 0)$ as a global minimizer and $f(x^*) = -0.4$. The relevant hypercube is $-1 \leq x_i \leq 1 \ \forall\, i \in \{1, ..., n\}$.

A.4 Easom Problem

The problem is

$$\min_x f(x) \ = -\cos(x_1)\cos(x_2)\exp\left(-(x_1 - \pi)^2 - (x_2 - \pi)^2\right)$$

with,

$$\frac{\partial f}{\partial x_1} = \cos(x_2)\exp\left(-(x_1 - \pi)^2 - (x_2 - \pi)^2(\sin(x_1) + 2(x_1 - \pi)\cos(x_1)\right)$$
$$\frac{\partial f}{\partial x_2} = \cos(x_1)\exp\left(-(x_1 - \pi)^2 - (x_2 - \pi)^2(\sin(x_2) + 2(x_2 - \pi)\cos(x_2)\right)$$

The dimension considered is $n = 2$, with $x^* = (\pi, \pi)$ as a global minimizer and $f(x^*) = -1$. The relevant hypercube is $-10 \leq x_1, x_2 \leq 10$.

A.5 Epistatic Michalewicz Problem

The problem is

$$\min_x f(x) \ = -\sum_{i=1}^{n} \sin(y_i)[\sin\left(i\frac{y_i^2}{\pi}\right)]^{2m}$$

where,

$$y_i = \begin{cases} x_i\cos(\theta) - x_{i+1}\sin(\theta) & \text{i odd} < n \\ x_i\sin(\theta) + x_{i+1}\cos(\theta) & \text{i even} < n \\ x_i & i = n \end{cases}$$

with,

$$\begin{aligned}
\frac{\partial f}{\partial x_i} &= -\frac{\partial y_i}{\partial x_i}\cos(y_i)[\sin\left(i\frac{y_i^2}{\pi}\right)]^{2m} - 4m\frac{iy_i}{\pi}\frac{\partial y_i}{\partial x_i}\sin(y_i)\cos\left(i\frac{y_i^2}{\pi}\right)[\sin\left(i\frac{y_i^2}{\pi}\right)]^{2m-1} \\
&\quad - \frac{\partial y_{i-1}}{\partial x_i}\cos(y_{i-1})[\sin\left((i-1)\frac{y_{i-1}^2}{\pi}\right)]^{2m} \\
&\quad - 4m\frac{(i-1)y_{i-1}}{\pi}\frac{\partial y_{i-1}}{\partial x_i}\sin(y_{i-1})\cos\left((i-1)\frac{y_{i-1}^2}{\pi}\right)[\sin\left((i-1)\frac{y_{i-1}^2}{\pi}\right)]^{2m-1}
\end{aligned}$$

$$\frac{\partial y_i}{\partial x_i} = \begin{cases} \cos(\theta) & \text{i odd} < n \\ \sin(\theta) & \text{i even} < n \\ 1 & i = n \end{cases}$$

$$\frac{\partial y_{i-1}}{\partial x_i} = \begin{cases} \cos(\theta) & \text{i odd} \\ -\sin(\theta) & \text{i even} \\ 0 & i = 1 \end{cases}$$

The dimension considered is $n = 10$, with

$$x^* = (2.693, 0.259, 2.074, 1.023, 2.275, 0.500, 2.138, 0.794, 2.219, 0.533)$$

as a global minimizer and $f(x^*) = -9.660152$. The relevant hypercube is $0 \leq x_i \leq \pi \ \forall\, i \in \{1, ..., n\}$.

A.6 Exponential Problem

The problem is

$$\min_x f(x) \ = -\exp\left(-0.5\sum_{i=1}^{n} x_i^2\right)$$

with,

$$\frac{\partial f}{\partial x_i} = x_i \exp\left(-0.5\sum_{i=1}^{n} x_i^2\right)$$

The dimension considered is $n = 10$, with $x^* = (0, ..., 0)$ as a global minimizer and $f(x^*) = -1$. The relevant hypercube is $-1 \leq x_i \leq 1 \ \forall\, i \in \{1, ..., n\}$.

A.7 Griewank Problem

The problem is

$$\min_x f(x) \ = 1 + \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right)$$

with,

$$\frac{\partial f}{\partial x_i} = \frac{x_i}{2000} + \frac{\sin\left(\frac{x_i}{\sqrt{i}}\right)}{\sqrt{i}} \prod_{j=1, j\neq i}^{n} \cos\left(\frac{x_j}{\sqrt{j}}\right)$$

The dimension considered is $n = 2$, with $x^* = (0, 0, ..., 0)$ as a global minimizer and $f(x^*) = 0$. The relevant hypercube is $-600 \leq x_i \leq 600 \ \forall\, i \in \{1, ..., n\}$.

A.8 Levy and Montalvo 1 Problem

The problem is

$$\min_x f(x) \ = \frac{\pi}{n}(10\sin^2(\pi y_1) + \sum_{i=1}^{n-1}(y_i - 1)^2[1 + 10\sin(\pi y_{i+1})^2] + (y_n - 1)^2)$$

where,

$$y_i = 1 + \frac{1}{4}(x_i + 1)$$

with,

$$\frac{\partial f}{\partial x_i} = \begin{cases} \frac{\pi}{n}[5\pi\sin(\pi y_1)\cos(\pi y_1) + \frac{1}{2}(y_1 - 1)(1 + 10\sin^2(\pi y_2))] & \text{if } i = 1 \\ \frac{\pi}{n}[\frac{1}{2}(y_n - 1) + 5\pi(y_{n-1} - 1)^2\sin(\pi y_n)\cos(\pi y_n)] & \text{if } i = n \\ \frac{\pi}{n}[\frac{1}{2}(y_i - 1)(1 + 10\sin^2(\pi y_{i+1})) + 5\pi(y_{i-1} - 1)^2\sin(\pi y_i)\cos(\pi y_i)] & \text{else} \end{cases}$$

The dimension considered is $n = 3$, with $x^* = (-1, -1, -1)$ as a global minimizer and $f(x^*) = 0$. The relevant hypercube is $-10 \leq x_i \leq 10 \ \forall\, i \in \{1, ..., n\}$.

A.9 Levy and Montalvo 2 Problem

The problem is

$$\min_x f(x) \ = 0.1(\sin^2(3\pi x_1) + \sum_{i=1}^{n-1}(x_i - 1)^2[1 + \sin^2(3\pi x_{i+1})] + (x_n - 1)^2[1 + \sin^2(2\pi x_n)])$$

with,

$$\frac{\partial f}{\partial x_i} = \begin{cases} 0.1[6\pi\sin(3\pi x_1)\cos(3\pi x_1) + 2(x_1 - 1)(1 + \sin^2(2\pi x_2))] & \text{if } i = 1 \\ 0.1[2(x_n - 1)(1 + \sin^2(2\pi x_n)) + 4\pi(x_n - 1)^2\sin(2\pi x_n)\cos(2\pi x_n)] & \text{if } i = n \\ \quad +6\pi(x_{n-1} - 1)^2\sin(3\pi x_n)\cos(3\pi x_n) & \\ 0.1[2(x_i - 1)(1 + \sin^2(3\pi x_{i+1})) + 6\pi(x_{i-1} - 1)^2\sin(2\pi x_i)\cos(3\pi x_i)] & \text{else} \end{cases}$$

The dimension considered is $n = 10$, with $x^* = (1, 1, 1, 1, 1, 1, 1, 1, 1, 1)$ as a global minimizer and $f(x^*) = 0$. The relevant hypercube is $-5 \leq x_i \leq 5 \ \forall\, i \in \{1, ..., n\}$.

A.10 Modified Langerman Problem

The problem is

$$\min_x f(x) \ = -\sum_{i=1}^{5} c_j \cos\left(\frac{d_j}{\pi}\right) \exp(-\pi d_j)$$

where,

$$d_j = \sum_{i=1}^{n} (x_i - a_{ji})^2$$

with,

$$\frac{\partial f}{\partial x_i} = \sum_{j=1}^{5} \frac{c_j}{\pi} \sin\left(\frac{d_j}{\pi}\right) \frac{\partial d_j}{\partial x_i} e^{-\pi d_j} + c_j \pi \cos\left(\frac{d_j}{\pi}\right) \frac{\partial d_j}{\partial x_i} e^{-\pi d_j}$$

$$\frac{\partial d_j}{\partial x_i} = 2(x_i - a_{ji})$$

The dimension considered is $n = 10$, with

$$x^* = (8.074, 8.777, 3.467, 1.867, 6.708, 6.349, 4.534, 0.276, 7.633, 1.567)$$

as a global minimizer and $f(x^*) = -0.965$. The data can be seen on A.3. The relevant hypercube is $0 \leq x_i \leq 10 \ \forall\, i \in \{1, ..., n\}$.

A.11 Neumaier 2 Problem

The problem is

$$\min_x f(x) \ = \sum_{k=1}^{n} (b_k - \sum_{i=1}^{n} x_i^k)^2$$

with,

$$\frac{\partial f}{\partial x_j} = \sum_{k=1}^{n} -2k x_j^{k-1} (b_k - \sum_{i=1}^{n} x_i^k)$$

The dimension considered is $n = 4$ and $b = (8, 18, 44, 114))$, with $x^* = (1, 2, 2, 3)$ as a global minimizer and $f(x^*) = 0$. The relevant hypercube is $0 \leq x_i \leq n \ \forall i \ \in \{1, ..., n\}$.

A.12 Neumaier 3 Problem

The problem is

$$\min_x f(x) \ = \sum_{i=1}^{n} (x_i - 1)^2 - \sum_{i=2}^{n} x_i x_{i-1}$$

with,

$$\frac{\partial f}{\partial x_i} = \begin{cases} 2(x_1 - 1) - x_2 & \text{if } i = 1 \\ 2(x_n - 1) - x_{n-1} & \text{if } i = n \\ 2(x_i - 1) - x_{i-1} - x_{i+1} & \text{else} \end{cases}$$

The dimension considered is $n = 10$, with $x_i^* = i(n + 1 - i)$ as a global minimizer and $f(x^*) = -\frac{n(n+4)(n-1)}{6}$. The relevant hypercube is $-n^2 \leq x_i \leq n^2 \ \forall\, i \in \{1, ..., n\}$.

A.13 Price's Transistor Modelling Problem

The problem is

$$\min_x f(x) \ = \gamma^2 + \sum_{k=1}^{4} (\alpha_k^2 + \beta_k^2)$$

where,

$$\alpha_k = (1 - x_1 x_2) x_3 [\exp(x_5(g_{1k} - g_{3k} x_7 10^{-3} - g_{5k} x_8 10^{-3})) - 1] - g_{5k} + g_{4k} x_2$$
$$\beta_k = (1 - x_1 x_2) x_4 [\exp(x_6(g_1 k - g_{2k} g_{3k} x_7 10^{-3} + g_{4k} x_9 10^{-3})) - 1] - g_{5k} x_1 + g_{4k}$$
$$\gamma = x_1 x_3 - x_2 x_4$$

with,

$$\frac{\partial f}{\partial x_i} = 2\gamma \frac{\partial \gamma}{\partial x_i} + \sum_{k=1}^{4} (2\alpha_k \frac{\partial \alpha_k}{\partial x_i} + 2\beta_k \frac{\partial \beta_k}{\partial x_i})$$

$$\frac{\partial \gamma}{\partial x_i} = \begin{cases} x_3 & i = 1 \\ -x_4 & i = 2 \\ x_1 & i = 3 \\ -x_3 & i = 4 \\ 0 & \text{else} \end{cases}$$

$$\frac{\partial \alpha_k}{\partial x_i} = \begin{cases} -x_2 x_3 [\exp(x_5(g_{1k} - g_{3k} x_7 10^{-3} - g_{5k} x_8 10^{-3})) - 1] & i = 1 \\ -x_1 x_3 [\exp(x_5(g_{1k} - g_{3k} x_7 10^{-3} - g_{5k} x_8 10^{-3})) - 1] + g_{4k} & i = 2 \\ (1 - x_1 x_2) [\exp(x_5(g_{1k} - g_{3k} x_7 10^{-3} - g_{5k} x_8 10^{-3})) - 1] & i = 3 \\ (1 - x_1 x_2) x_3 [g_{1k} - g_{3k} x_7 10^{-3} - g_{5k} x_8 10^{-3}] & i = 5 \\ \exp[x_5(g_{1k} - g_{3k} x_7 10^{-3} - g_{5k} x_8 10^{-3})] & \\ -(1 - x_1 x_2) x_3 g_{3k} x_5 10^{-3} \exp(x_5(g_{1k} - g_{3k} x_7 10^{-3} - g_{5k} x_8 10^{-3})) & i = 7 \\ -(1 - x_1 x_2) x_3 g_{5k} x_5 10^{-3} \exp(x_5(g_{1k} - g_{3k} x_7 10^{-3} - g_{5k} x_8 10^{-3})) & i = 8 \\ 0 & \text{else} \end{cases}$$

$$\frac{\partial \beta_k}{\partial x_i} = \begin{cases} -x_2 x_4 [\exp(x_6(g_1 k - g_{2k} g_{3k} x_7 10^{-3} + g_{4k} x_9 10^{-3})) - 1] - g_{5k} & i = 1 \\ -x_1 x_4 [\exp(x_6(g_1 k - g_{2k} g_{3k} x_7 10^{-3} + g_{4k} x_9 10^{-3})) - 1] & i = 2 \\ (1 - x_1 x_2) [\exp(x_6(g_1 k - g_{2k} g_{3k} x_7 10^{-3} + g_{4k} x_9 10^{-3})) - 1] & i = 4 \\ (1 - x_1 x_2) x_4 (g_1 k - g_{2k} g_{3k} x_7 10^{-3} + g_{4k} x_9 10^{-3}) & i = 6 \\ \exp(x_6(g_1 k - g_{2k} g_{3k} x_7 10^{-3} + g_{4k} x_9 10^{-3})) & \\ -(1 - x_1 x_2) x_4 g_{3k} x_6 10^{-3} \exp(x_6(g_1 k - g_{2k} - g_{3k} x_7 10^{-3} + g_{4k} x_9 10^{-3})) & i = 7 \\ (1 - x_1 x_2) x_4 g_{4k} x_6 10^{-3} \exp(x_6(g_1 k - g_{2k} - g_{3k} x_7 10^{-3} + g_{4k} x_9 10^{-3})) & i = 9 \\ 0 & \text{else} \end{cases}$$

with $x^* = (0.9, 0.45, 1, 2, 8, 8, 5, 1, 2)$ as a global minimizer and $f(x^*) = 0$. The relevant hypercube is $-10 \leq x_i \leq 10 \ \forall\, i \in \{1, ..., 9\}$.

Table A.2 PTM data from [21]

i	g_{ik}			
	k = 1	2	3	4
1	0.485	0.752	0.869	0.982
2	0.369	1.254	0.703	1.455
3	5.2095	10.0677	22.9274	20.2153
4	23.3037	101.779	111.461	191.267
5	28.5132	111.8467	134.3884	211.4823

A.14 Rastrigin Problem

The problem is

$$\min_x f(x) \ = 10n + \sum_{i=1}^{n} (x_i^2 - 10\cos(2\pi x_i))$$

with,

$$\frac{\partial f}{\partial x_i} = 2x_i + 20\pi \sin(2\pi x_i)$$

The dimension considered is $n = 10$, with $x_i^* = (0, 0, .., 0)$ as a global minimizer and $f(x^*) = 0$. The relevant hypercube is $-5.12 \leq x_i \leq 5.12 \ \forall\, i \in \{1, ..., n\}$.

A.15 Schaffer 1 Problem

The problem is

$$\min_x f(x) \ = 0.5 + \frac{\left(\sin\left(\sqrt{x_1^2 + x_2^2}\right)\right)^2 - 0.5}{(1 + 0.001(x_1^2 + x_2^2))^2}$$

with,

$$s = x_1^2 + x_2^2$$

$$\frac{\partial f}{\partial x_i} = x_i \frac{2(\sin\sqrt{s})(\cos\sqrt{s})s^{-0.5}(1 + 0.001s) + 0.004\left(\sin\sqrt{s}\right)^2 - 0.5)}{(1 + 0.001s)^4}$$

The dimension considered is $n = 2$, with $x_i^* = (0, 0)$ as a global minimizer and $f(x^*) = 0$. The relevant hypercube is $-100 \leq x_1, x_2 \leq 100$.

A.16 Schaffer 2 Problem

The problem is

$$\min_x f(x) \ = (x_1^2 + x_2^2)^{0.25}(\sin^2(50(x_1^2 + x_2^2)^{0.1}) + 1)$$

with,

$$\begin{aligned} s &= x_1^2 + x_2^2 \\ \frac{\partial f}{\partial x_i} &= x_i[0.5s^{-0.75}(\sin^2(50s^{0.1}) + 1) + 20s^{-0.65}\sin(50s^{0.1})\cos(50s^{0.1})] \end{aligned}$$

The dimension considered is $n = 2$, with $x_i^* = (0, 0)$ as a global minimizer and $f(x^*) = 0$. The relevant hypercube is $-100 \leq x_1, x_2 \leq 100$.

A.17 Shekel's Foxholes Problem

The problem is

$$\min_x f(x) \ = -\sum_{j=1}^{30} \frac{1}{c_j + \sum_{i=1}^{n}(x_i - a_{ji})^2}$$

with,

$$\frac{\partial f}{\partial x_i} = \sum_{j=1}^{30} \frac{2(x_i - a_{ji})}{(c_j + \sum_{i=1}^{n}(x_i - a_{ji})^2)^2}$$

The dimension considered is $n = 10$, with

$$x^* \approx (8.025, 9.152, 5.114, 7.621, 4.564, 4.771, 2.996, 6.126, 0.734, 4.982)$$

as a global minimizer and $f(x^*) = -10.208792792153845$. The data can be seen in figure A.3. The relevant hypercube is $0 \leq x_i \leq 10 \ \forall\, i \in \{1, ..., n\}$.

Table A.3 ML and FX data from [21]

j	c_j	a_{ji}									
		i = 1	2	3	4	5	6	7	8	9	10
1	0.806	9.681	0.667	4.783	9.095	3.517	9.325	6.544	0.211	5.122	2.020
2	0.517	9.400	2.041	3.788	7.931	2.882	2.672	3.568	1.284	7.033	7.374
3	0.100	8.025	9.152	5.114	7.621	4.564	4.711	2.996	6.126	0.734	4.982
4	0.908	2.196	0.415	5.649	6.979	9.510	9.166	6.304	6.054	9.377	1.426
5	0.965	8.074	8.777	3.467	1.863	6.708	6.349	4.534	0.276	7.633	1.567
6	0.669	7.650	5.658	0.720	2.764	3.278	5.283	7.474	6.274	1.409	8.208
7	0.524	1.256	3.605	8.623	6.905	4.584	8.133	6.071	6.888	4.187	5.448
8	0.902	8.314	2.261	4.224	1.781	4.124	0.932	8.129	8.658	1.208	5.762
9	0.531	0.226	8.858	1.420	0.945	1.622	4.698	6.228	9.096	0.972	7.637
10	0.876	7.305	2.228	1.242	5.928	9.133	1.826	4.060	5.204	8.713	8.247
11	0.462	0.652	7.027	0.508	4.876	8.807	4.632	5.808	6.937	3.291	7.016
12	0.491	2.699	3.516	5.874	4.119	4.461	7.496	8.817	0.690	6.593	9.789
13	0.463	8.327	3.897	2.017	9.570	9.825	1.150	1.395	3.885	6.354	0.109
14	0.714	2.132	7.006	7.136	2.641	1.882	5.943	7.273	7.691	2.880	0.564
15	0.352	4.707	5.579	4.080	0.581	9.698	8.542	8.077	8.515	9.231	4.670
16	0.869	8.304	7.559	8.567	0.322	7.128	8.392	1.472	8.524	2.277	7.826
17	0.813	8.632	4.409	4.832	5.768	7.050	6.715	1.711	4.323	4.405	4.591
18	0.811	4.887	9.112	0.170	8.967	9.693	9.867	7.508	7.770	8.382	6.740
19	0.828	2.440	6.686	4.299	1.007	7.008	1.427	9.398	8.480	9.950	1.675
20	0.964	6.306	8.583	6.084	1.138	4.350	3.134	7.853	6.061	7.457	2.258
21	0.789	0.652	2.343	1.370	0.821	1.310	1.063	0.689	8.819	8.833	9.070
22	0.360	5.558	1.272	6.756	9.857	2.279	2.764	1.284	1.677	1.244	1.234
23	0.369	3.352	7.549	9.817	9.437	8.687	4.167	2.570	6.540	0.228	0.027
24	0.992	8.798	0.880	2.370	0.168	1.701	3.680	1.231	2.390	2.499	0.064
25	0.332	1.460	8.057	1.336	7.217	7.914	3.615	9.981	9.198	5.292	1.224
26	0.817	0.432	8.654	8.774	0.249	8.801	7.461	4.416	0.652	4.002	4.644
27	0.632	0.679	2.800	5.523	3.049	2.968	7.225	6.730	4.199	9.614	9.229
28	0.883	4.263	1.074	7.286	5.599	8.291	5.200	9.214	8.272	4.398	4.506
29	0.608	9.496	4.830	3.150	8.270	5.079	1.231	5.731	9.494	1.883	9.732
30	0.326	4.138	2.562	2.532	9.661	5.611	5.500	6.886	2.341	9.699	6.500

A.18 Shubert Problem

The problem is

$$\min_x f(x) \ = \prod_{i=1}^{n} (\sum_{j=1}^{5} j \cos((j+1)x_i + j))$$

with,

$$\frac{\partial f}{\partial x_i} = -(\sum_{j=1}^{5} j(j+1)\sin((j+1)x_i + j)) \prod_{k=1, k\neq i}^{n} (\sum_{j=1}^{5} j\cos((j+1)x_i + j))$$

The dimension considered is $n = 2$, with 18 global minimizers in two dimension and $f(x^*) = -186,7309$ (186.7309088310238). The relevant hypercube is $-10 \leq x_i \leq 10 \ \forall\, i \in \{1, ..., n\}$.

A.19 Sinusoidal Problem

The problem is

$$\min_x f(x) \ = -(A\prod_{i=1}^{n}\sin(x_i - z) + \prod_{i=1}^{n}\sin(B(x_i - z)))$$

with,

$$\frac{\partial f}{\partial x_i} = -(A\cos(x_i - z)\prod_{k=1,k\neq i}^{n}\sin(x_i - z) + B\cos(B(x_i - z))\prod_{k=1,k\neq i}^{n}\sin(B(x_i - z)))$$

The dimension considered is $n = 10$, the values of the constants chosen for the test are $A = 2.5$, $B = 5$, and $z = 30$. We have $x^* \approx (90 + z, 90 + Z, ..., 90 + Z)$ as a global minimizer and $f(x^*) = -(A + 1)$. The relevant hypercube is $0 \leq x_i \leq 180 \ \forall\, i \in \{1, ..., n\}$.

A.20 Storn's Tchebychev Problem

The problem is

$$\min_x f(x) \ = p_1(x) + p_2(x) + p_3(x)$$

where we define,

$$p_1(x) = \begin{cases} (u-d)^2 & \text{if } u < d \\ 0 & \text{else.} \end{cases} \qquad u = \sum_{i=1}^{n}(1.2)^{n-i}x_i$$

$$p_2(x) = \begin{cases} (v-d)^2 & \text{if } v < d \\ 0 & \text{else.} \end{cases} \qquad v = \sum_{i=1}^{n}(-1.2)^{n-i}x_i$$

$$p_3(x) = \sum_{k=1}^{m} l_k, \ \ l_k = \begin{cases} (w_k-1)^2 & \text{if } w_k > 1 \\ (w_k+1)^2 & \text{if } w_k < 1 \\ 0 & \text{else.} \end{cases} \qquad w_k = \sum_{i=1}^{n}\left(\frac{2k}{m} - 1\right)^{n-i} x_i$$

with,

$$\frac{\partial f}{\partial x_j} = \frac{\partial p_1(x)}{\partial x_j} + \frac{\partial p_3(x)}{\partial x_j} + \frac{\partial p_3(x)}{\partial x_j}$$

$$\frac{\partial p_1(x)}{\partial x_j} = \begin{cases} 2(u-d)1.2^{n_j} & \text{if } u < d \\ 0 & \text{else.} \end{cases}$$

$$\frac{\partial p_2(x)}{\partial x_j} = \begin{cases} 2(v-d)(-1.2)^{n_j} & \text{if } v < d \\ 0 & \text{else.} \end{cases}$$

$$\frac{\partial p_3(x)}{\partial x_j} = \sum_{k=1}^{m} l_k, \ \ l_k = \begin{cases} 2(w_k-1)\left(\frac{2k}{m} - 1\right)^{n-j} & \text{if } w_k > 1 \\ 2(w_k+1)\left(\frac{2k}{m} - 1\right)^{n-j} & \text{if } w_k < 1 \\ 0 & \text{else.} \end{cases}$$

The dimension considered is $n = 9$, with

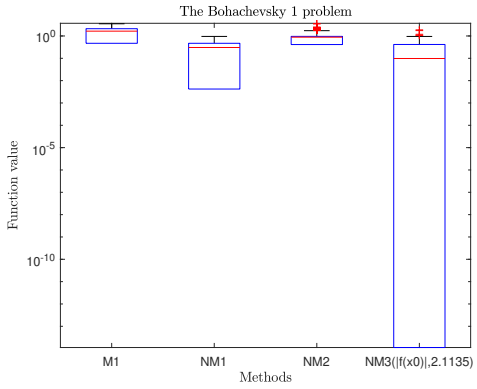
$$x^* \approx (128, 0, -256, 0, 160, 0, -32, 0, 1)$$

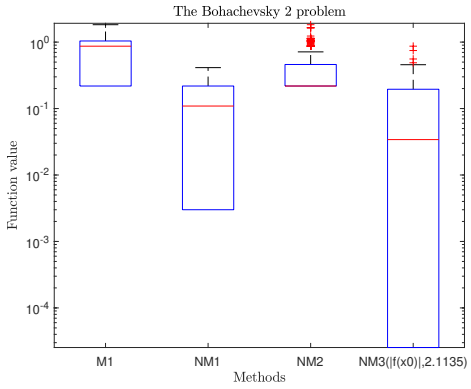
as a global minimizer and $f(x^*) = 0$. The relevant hypercube is $-128 \leq x_i \leq 128 \ \forall \, i \in \{1, ..., n\}$.

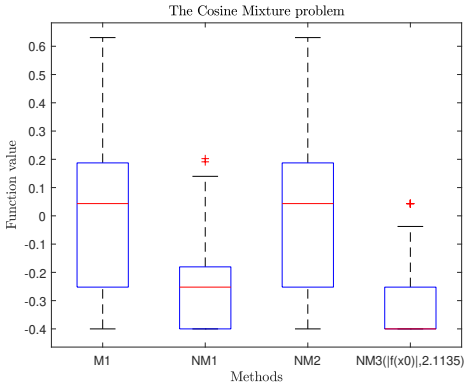
Appendix B

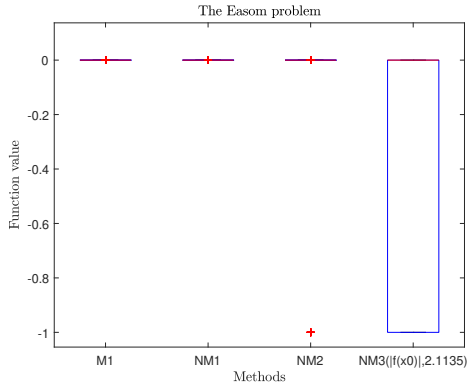
Box plot analysis

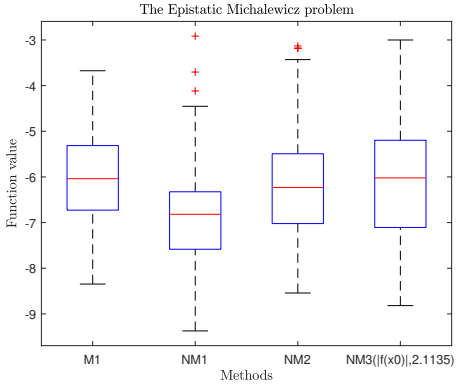
B.1 Comparison between the algorithms

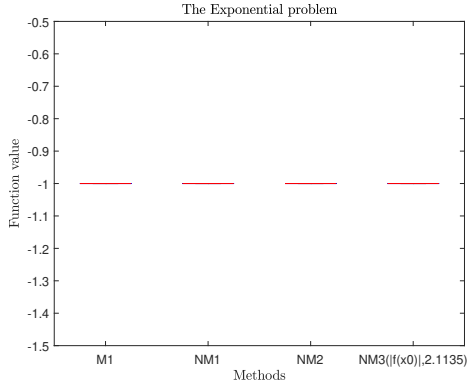


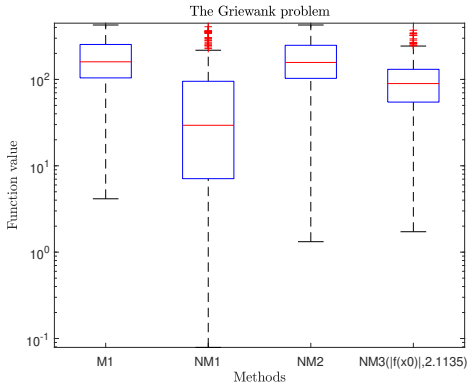


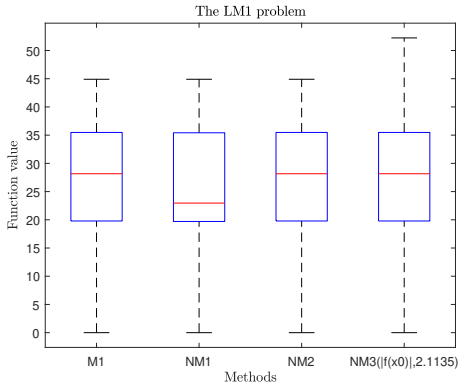


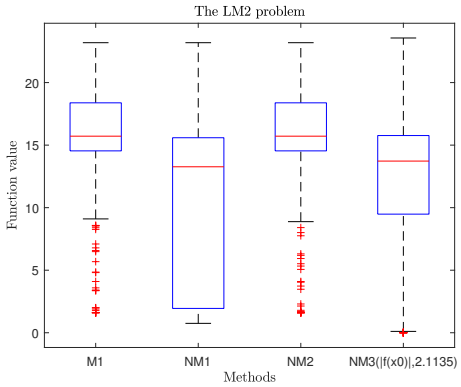


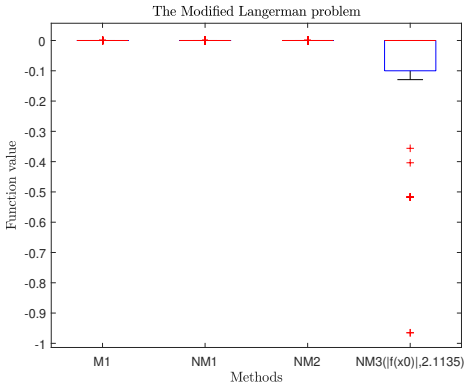


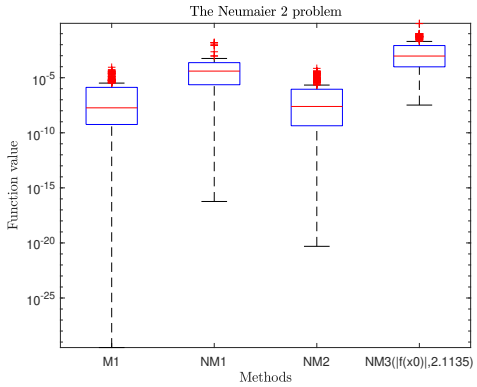


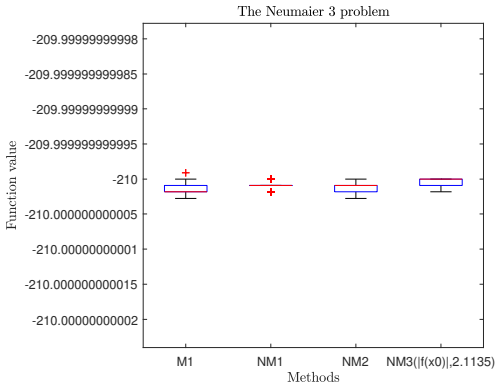

The Griewank problem
Function value
Methods
M1
NM1
NM2
NM3(|f(x0)|,2.1135)

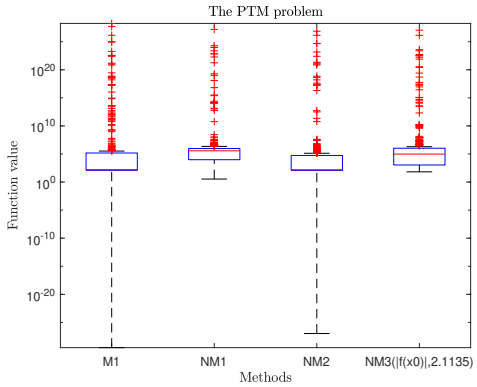

The LM1 problem
Function value
Methods
M1
NM1
NM2
NM3(|f(x0)|,2.1135)

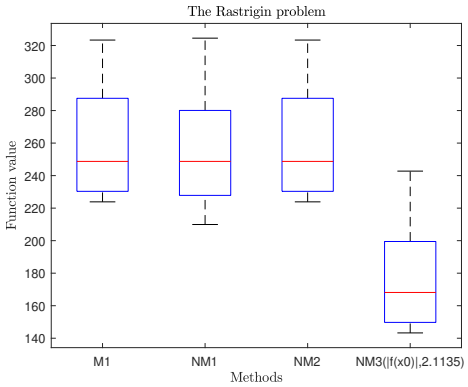

The LM2 problem
Function value
Methods
M1
NM1
NM2
NM3(|f(x0)|,2.1135)

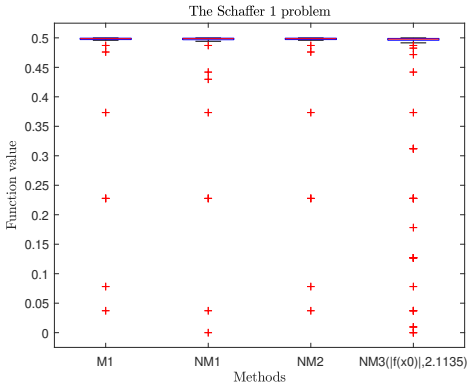

The Modified Langerman problem
Function value
Methods
M1
NM1
NM2
NM3(|f(x0)|,2.1135)

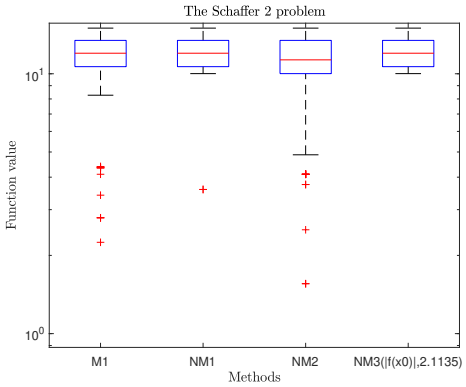

The Neumaier 2 problem
Function value
Methods
M1
NM1
NM2
NM3(|f(x0)|,2.1135)

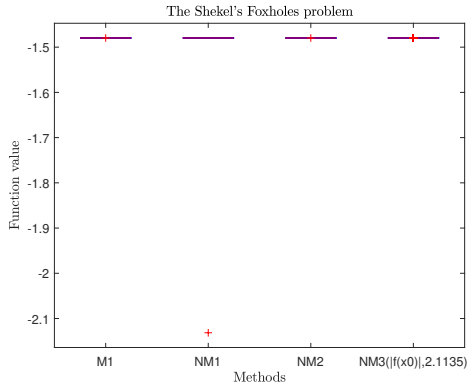

The Neumaier 3 problem
Function value
Methods
M1
NM1
NM2
NM3(|f(x0)|,2.1135)

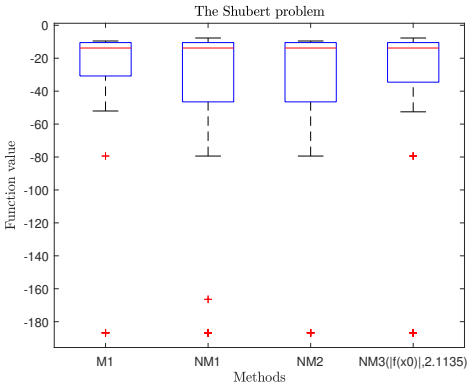

The PTM problem
Function value
Methods
M1
NM1
NM2
NM3(|f(x0)|,2.1135)

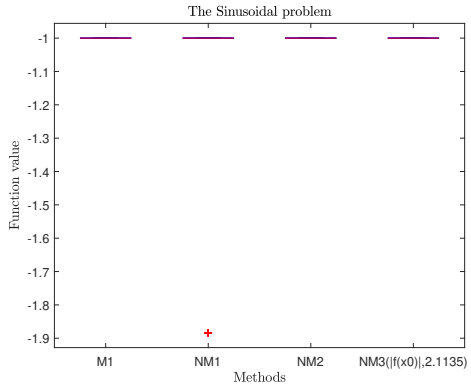

The Rastrigin problem
Function value
Methods
M1
NM1
NM2
NM3(|f(x0)|,2.1135)

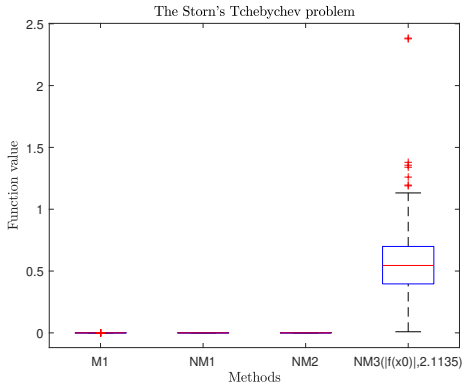

The Schaffer 1 problem
Function value
0.5
0.45
0.4
0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
M1
NM1
NM2
NM3(|f(x0)|,2.1135)
Methods


The Schaffer 2 problem
Function value
10^1
10^0
M1
NM1
NM2
NM3(|f(x0)|,2.1135)
Methods

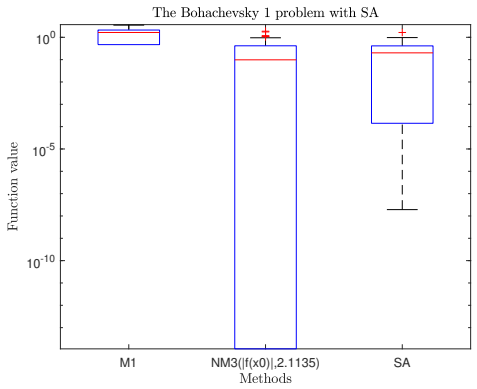

The Shekel's Foxholes problem
Function value
-1.5
-1.6
-1.7
-1.8
-1.9
-2
-2.1
M1
NM1
NM2
NM3(|f(x0)|,2.1135)
Methods

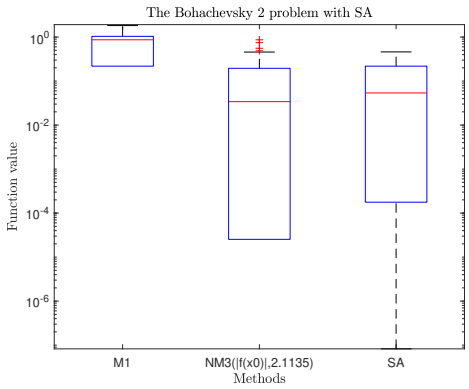

The Shubert problem
Function value
0
-20
-40
-60
-80
-100
-120
-140
-160
-180
M1
NM1
NM2
NM3(|f(x0)|,2.1135)
Methods

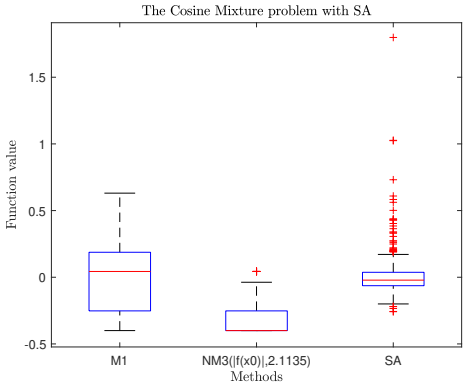

The Sinusoidal problem
Function value
-1
-1.1
-1.2
-1.3
-1.4
-1.5
-1.6
-1.7
-1.8
-1.9
M1
NM1
NM2
NM3(|f(x0)|,2.1135)
Methods

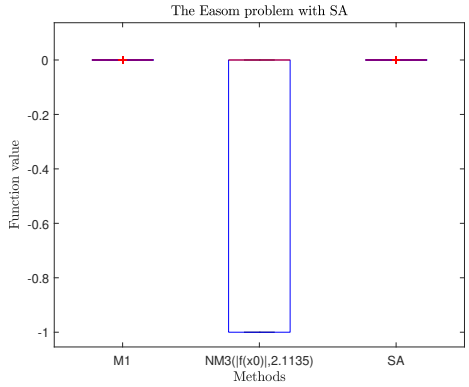

The Storn's Tchebychev problem
Function value
2.5
2
1.5
1
0.5
0
M1
NM1
NM2
NM3(|f(x0)|,2.1135)
Methods

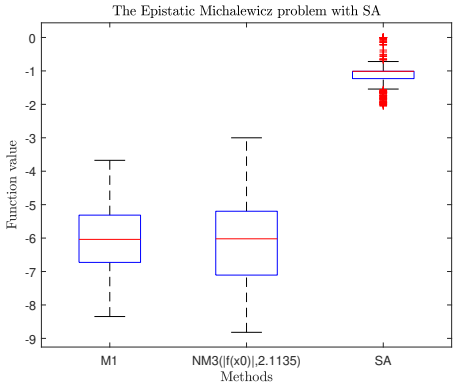
B.2 Comparison with the simulated annealing

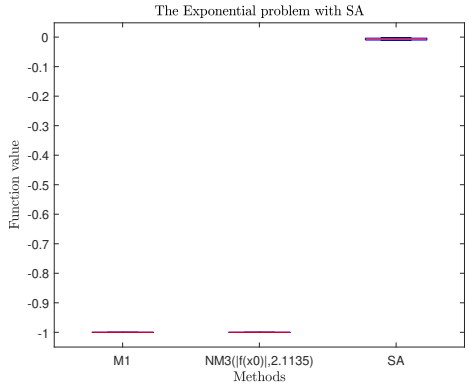


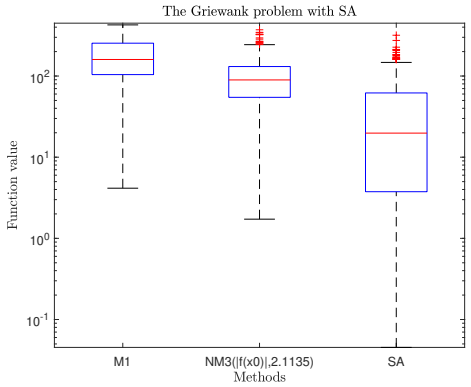


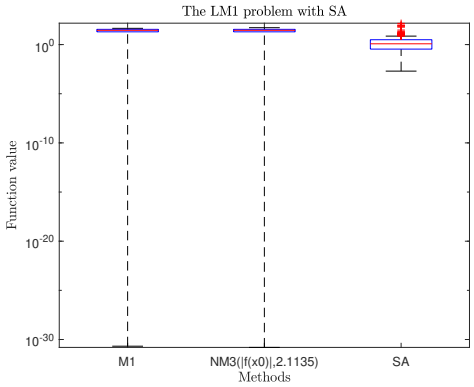


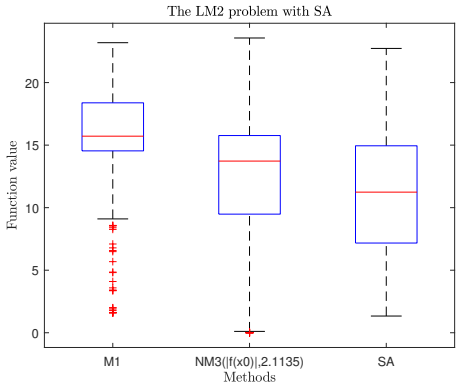


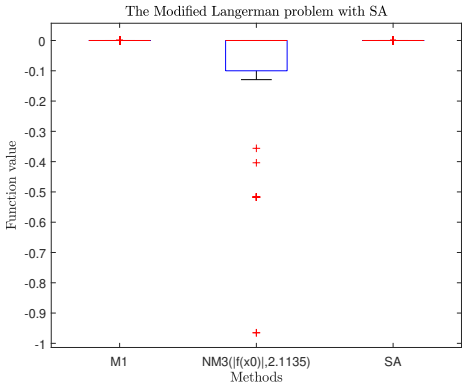


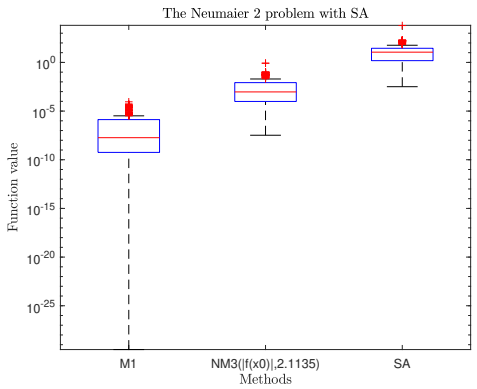


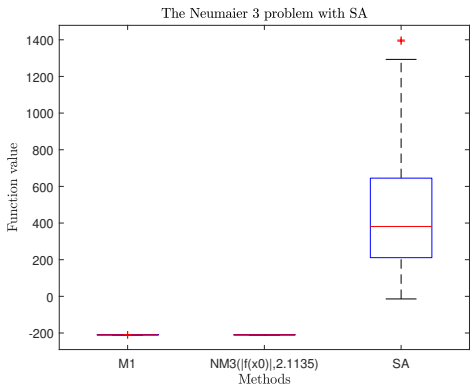

The Griewank problem with SA
Function value
M1
NM3(|f(x0)|,2.1135)
SA
Methods

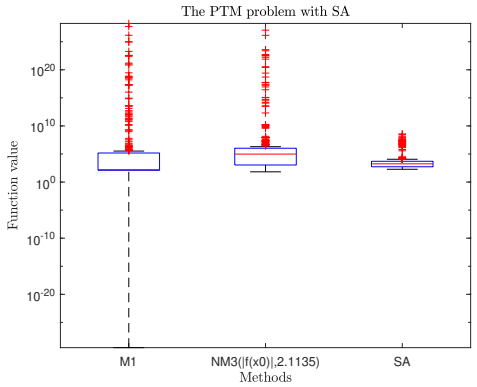

The LM1 problem with SA
Function value
M1
NM3(|f(x0)|,2.1135)
SA
Methods

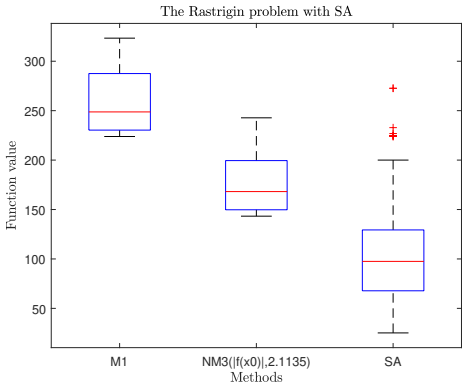

The LM2 problem with SA
Function value
M1
NM3(|f(x0)|,2.1135)
SA
Methods

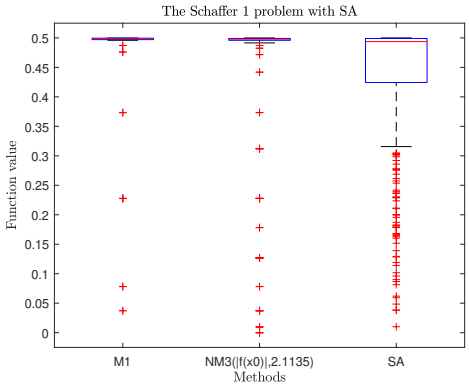

The Modified Langerman problem with SA
Function value
M1
NM3(|f(x0)|,2.1135)
SA
Methods

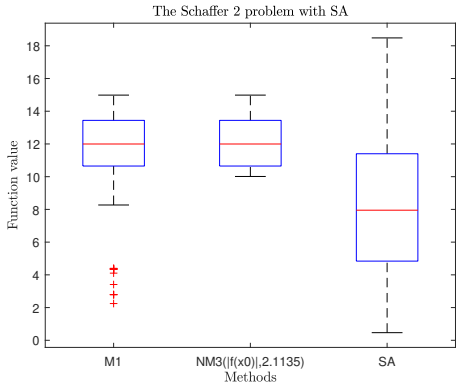

The Neumaier 2 problem with SA
Function value
M1
NM3(|f(x0)|,2.1135)
SA
Methods

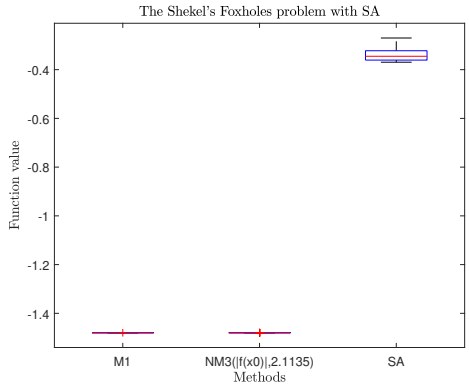

The Neumaier 3 problem with SA
Function value
M1
NM3(|f(x0)|,2.1135)
SA
Methods

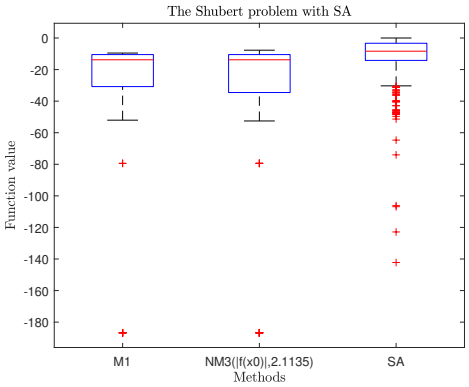

The PTM problem with SA
Function value
M1
NM3(|f(x0)|,2.1135)
SA
Methods

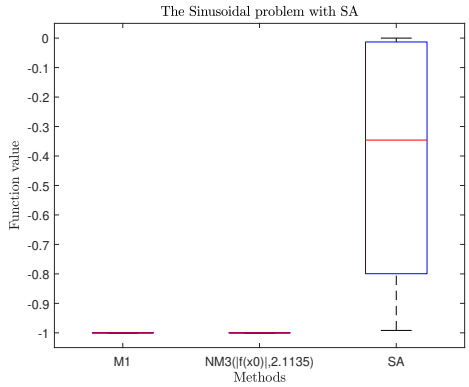

The Rastrigin problem with SA
Function value
M1
NM3(|f(x0)|,2.1135)
SA
Methods

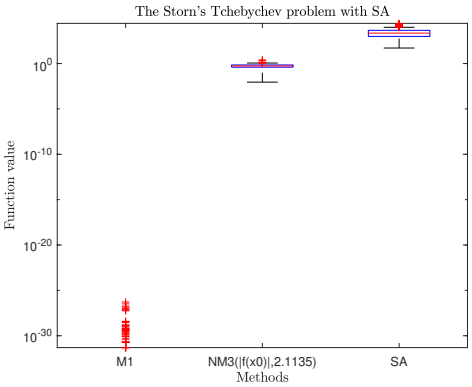

The Schaffer 1 problem with SA
Function value
0.5
0.45
0.4
0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
M1
NM3(|f(x0)|,2.1135)
SA
Methods


The Schaffer 2 problem with SA
Function value
18
16
14
12
10
8
6
4
2
0
M1
NM3(|f(x0)|,2.1135)
SA
Methods


The Shekel's Foxholes problem with SA
Function value
-0.4
-0.6
-0.8
-1
-1.2
-1.4
M1
NM3(|f(x0)|,2.1135)
SA
Methods


The Shubert problem with SA
Function value
0
-20
-40
-60
-80
-100
-120
-140
-160
-180
M1
NM3(|f(x0)|,2.1135)
SA
Methods


The Sinusoidal problem with SA
Function value
0
-0.1
-0.2
-0.3
-0.4
-0.5
-0.6
-0.7
-0.8
-0.9
-1
M1
NM3(|f(x0)|,2.1135)
SA
Methods


The Storn's Tchebychev problem with SA
Function value
10^0
10^-10
10^-20
10^-30
M1
NM3(|f(x0)|,2.1135)
SA
Methods

UNIVERSITÉ CATHOLIQUE DE LOUVAIN
École polytechnique de Louvain
Rue Archimède, 1 bte L6.11.01, 1348 Louvain-la-Neuve, Belgique | **www.uclouvain.be/epl**