UCLouvain

École polytechnique de Louvain

Derivative-free method for solving underdetermined systems of nonlinear equations

Author: **Timothé Taminiau**
Supervisor: **Geovani Nunes Grapiglia**
Readers: **Estelle Massart, Julien Hendrickx**
Academic year 2022–2023
Master [120] in Mathematical Engineering

Abstract

In this thesis, we study derivative-free methods able to solve systems of nonlinear equations with a particular emphasis on the underdetermined case. We present a new class of methods which generalizes the works from Grapiglia and Chorobura [11] and Echebest, Schuverdt, and Vignau [9]. If the Jacobian corresponding to the system of nonlinear equations is Lipschitz continuous then we prove, for two particular methods belonging to this class, worst-case complexity bounds on the number of function evaluations to reach some optimality measure. Afterwards, the methods are implemented and tested numerically to observe how they behave in practice. Finally, we provide an application of these methods in the context of black-box attack against deep neural networks. In particular, we model mathematically the way to modify slightly an input such that it is misclassified with the constraint that the neural network's structure is hidden.

Keywords : Underdetermined systems of nonlinear equations · Derivative-free · Nonmonotone line search · Complexity bounds · Black-box adversarial attack

Acknowledgements

Firstly, I am extremely grateful to my supervisor, Professor Geovani Nunes Grapiglia, for all our meetings, for the numerous suggestions, for answering my questions and for guiding me through this interesting subject of thesis.

Secondly, I am also thankful to the members of my jury, Professors Estelle Massart and Julien Hendrickx who accepted to examine and evaluate my work.

Finally, I would like to acknowledge my parents and my sister for their feedbacks and their support.

Contents

Chapter 1

State of the art

This chapter is mostly about reviewing some notions, motivating the topics of this thesis and giving a brief overview of the existing methods for solving systems of nonlinear equations.

1.1 Background

In this thesis, we are interested in solving the system of nonlinear equations

$$F(x) = 0$$

where the mapping $F : \mathbb{R}^n \to \mathbb{R}^m$ is continuously differentiable with $n \geq m$. The Jacobian matrix of $F(x)$ will be noted $J(x)$. Moreover, we might also need to define a merit function $f : \mathbb{R}^n \to \mathbb{R}_+$ corresponding to F such that we have $F(x) = 0$ for some $x \in \mathbb{R}^n$ if and only if $f(x) = 0$. From this definition, it is clear that when the system of equations admits at least one solution, we have

$$\min_{x \in \mathbb{R}^n} f(x) \quad \Longleftrightarrow \quad \text{Find } x \in \mathbb{R}^n \text{ such that } F(x) = 0$$

Hence, we will often see the algorithms used to solve systems of nonlinear equations as a way to minimize f. Nevertheless if f doesn't reach 0 at some point then such interpretation is no longer possible and the set $\{x \in \mathbb{R}^n \mid F(x) = 0\}$ must be empty. We focus on a merit function of the form

$$f(x) = \frac{1}{2}||F(x)||^2 \tag{1.1}$$

where we use the notation $||.||$ for the usual 2-norm namely $||x|| = \sqrt{\sum_{i=1}^n x_i^2}$ for any $x \in \mathbb{R}^n$ and corresponds to the induced 2-norm for matrices. Although, we might also mention other possible merit functions such as $f(x) = ||F(x)||^p$ for $p \in \{1,\ 2\}$ used in Cruz, Martínez, and Raydan [6]. An advantage of the merit function (1.1) is that f is also continuously differentiable and its gradient has a concise formula given by

$$\nabla f(x) = J(x)^T F(x)$$

Contrary to the framework of optimization methods, finding a point $\bar{x}$ which is almost stationary meaning $\nabla f(\bar{x}) = J(\bar{x})^T F(\bar{x}) \approx 0$ is not always a good thing. Indeed, we can converge to a stationary point $\bar{x}$ that may be only a local minimum hence with $F(\bar{x}) \neq 0$. However, under additional hypothesis on the structure of the Jacobian, we should be able to go further into the developments. For example, we can observe that if $\text{Ker}(J(x)^T) = \{0\}$ for all $x \in \mathbb{R}^n$ then $\nabla f(x) = 0$ if and only if $F(x) = 0$.

Thereafter, we recall few definitions of the properties that we can expect from the mapping $F(x)$ and that may be useful in the *Chapter 2*.

The Jacobian $J : \mathbb{R}^n \to \mathbb{R}^{m\times n}$ is said Lipschitz continuous with constant $L_J > 0$ if

$$||J(x) - J(y)|| \leq L_J||x - y|| \tag{1.2}$$

The hypothesis of Lipschitz continuity is quite common when we study the theoretical convergence of methods. Roughly speaking, inequality (1.2) means that the rate of change of the Jacobian is bounded by some constant. In addition, we can also consider mappings $F : \mathbb{R}^n \to \mathbb{R}^n$ that are μ-strongly monotone that is to say there exists some $\mu > 0$ such that

$$(F(x) - F(y))^T(x - y) \geq \mu||x - y||^2 \quad \forall x, y \in \mathbb{R}^n \tag{1.3}$$

As explained in Geiger and Kanzow [10], this property is stronger than the monotonicity for a multivariable function $F(x)$ defined as

$$(F(x) - F(y))^T(x - y) \geq 0 \quad \forall x, y \in \mathbb{R}^n$$

and it can be shown that (1.3) is equivalent to have for any $x \in \mathbb{R}^n$

$$v^T J(x)v \geq \mu||v||^2 \quad \forall v \in \mathbb{R}^n$$

Notice that the class of problems with μ-strongly monotone mapping is actually larger than the previous definitions which is actually only true for strongly increasing mapping. Indeed, if $G : \mathbb{R}^n \to \mathbb{R}^n$ is a μ-strongly decreasing mapping then there exists $\mu > 0$ such that

$$-(G(x) - G(y))^T(x - y) \geq \mu||x - y||^2$$

Therefore, $-G(x)$ is μ-strongly monotone and we can reformulate without loss of generality the system of nonlinear equations as

$$F(x) = -G(x) = 0$$

1.2 Motivations

In many fields of Applied Mathematics, it is common to come across the problem of finding $x \in \mathbb{R}^n$ such that

$$F(x) = Ax - b = 0 \tag{1.4}$$

where $A \in \mathbb{R}^{m\times n}$ and $b \in \mathbb{R}^m$. The equation (1.4) is known as a system of linear equations and it is a quite well understood problem which can always be solved exactly in a fixed number of steps thanks to Gaussian elimination. Firstly, linear system may appear in approximation theory when we want to interpolate a given set of points with a polynomial. Secondly, being able to solve a linear system is also useful when we would like to compute numerically the solution of partial differential equations. Indeed, the discretization of a linear partial differential equation with finite differences or finite elements results generally in a large scale linear system. Thirdly, we might also want to do quadratic unconstrained minimization namely

$$\min_{x\in\mathbb{R}^n} \frac{1}{2}x^T Ax - b^T x \tag{1.5}$$

where A is positive semi-definite. Note that (1.4) and (1.5) have the same solutions.

However, the real world is mostly nonlinear and many problems can't be written under the form (1.4). For example, we might interpolate within a set of functions which are parameterized in a non trivial way, discretize a differential equation presenting nonlinearity or minimize an arbitrary function. All these problems motivate the need for designing efficient algorithms able to solve systems of nonlinear equations.

We give thereafter few definitions which are really the key concepts to understand the methods presented in this work

- ✧ **Derivative-free :** A method for solving the problem $F(x) = 0$ is called derivative-free if it performs without any information about the derivatives of F hence we are only allowed to query the value $F(x)$ for any given point $x \in \mathbb{R}^n$.
- ✧ **Nonmonotone :** A method for solving the problem $F(x) = 0$ has a nonmonotone behavior if the sequence of iterates $(x_k)_{k=0}^{\infty}$ generated by the method is such that $||F(x_k)||$ is not always decreasing.
- ✧ **Underdetermined :** A system of nonlinear equations of the form $F(x) = 0$ with $F : \mathbb{R}^n \to \mathbb{R}^m$ is underdetermined if there is (strictly) more unknowns than equations meaning $n > m$. Oppositely, if $n = m$ then the system of equations is said determined.

Underdetermined systems of equations typically arise in the context of constrained optimization where we want to solve

$$\min_{x \in \mathbb{R}^n} \; g(x) \quad \text{such that} \quad c(x) = 0 \tag{1.6}$$

with $g : \mathbb{R}^n \to \mathbb{R}$ the objective function and $c : \mathbb{R}^n \to \mathbb{R}^m$ the function (potentially very complex) describing the constraints. Typically, we have an optimization method which can solve (1.6) by generating a sequence of iterates that remains in the admissible set. However, the method needs to start with an admissible point x_0. It corresponds to finding $x_0 \in \mathbb{R}^n$ such that $c(x_0) = 0$ which is an underdetermined system of nonlinear equations because problems like (1.6) admit in general more variables than constraints.

For a long time, the nonmonotone methods haven't been used because the possibility to observe $(f(x_k))_{k=0}^{\infty}$ with significant increases between some iterates was a bit disconcerting. Indeed, the majority of theorems about global convergence required a sufficient decrease condition on the objective function at the next step. Actually, methods usually produce a first guess in order to find the next iterate x_{k+1}. It often happens that this guess violates the monotonicity condition but contains in general sensibly more information about the structure of the problem. Therefore, the nonmonotone approach tends to be more prone to accept the first guess instead of modifying it so that the objective decreases enough.

The derivative-free methods are based only on the information provided by function values and are not allowed to access to the Jacobian. Notice that the evaluation of the function $F(x)$ is generally much more cheaper than its Jacobian $J(x)$. Therefore, the computation and then the storage in memory of a matrix with size $m \times n$

might become difficult for large values of m or/and n. In numerous fields of Science such as Physics, Chemistry, Statistics, ... the data come from experiments and observations hence the computation of formal derivatives concerning theses quantities is not possible in practice. On one hand, real systems are often very complex to model exactly thus the derivative-free setup allows to treat these systems as black-box giving function value as output for any input. On the other hand, derivative-free methods make barely no explicit assumptions on the problem, while methods using the Jacobian will require F to be at least differentiable.

A practical application of derivative-free methods is the design of particular inputs able to mislead a classifier called adversarial examples. Especially in the context of deep neural networks, adversarial examples are usually made without any information about the structure or the weights. We are only allowed to give an input to the classifier and then observe the corresponding output. This application is explained in full details in Chapter 4.

1.3 Newton method

The most famous method used to solve the problem $F(x) = 0$ is certainly the Newton method which is reminded below

Algorithm 1: Newton method	Suitable for $n \geq m$
Data: Starting point $x_0 \in \mathbb{R}^n$	
Step 0. Set $k = 0$	
Step 1. Solve $J(x_k)d_k = -F(x_k)$	
Step 2. Pick a step size $\alpha_k > 0$	
Step 3. $x_{k+1} = x_k + \alpha_k d_k$	
Step 4. Do $k \leftarrow k+1$ and go back to Step 1.	

The method can be retrieved heuristically with the following reasoning :

- ✧ Let x_k be a starting point not too far from the solution $x^\star$
- ✧ Assume F is approximately linear in the neighbourhood of x_k then

$$F(x_{k+1}) \approx F(x_k) + J(x_k)(x_{k+1} - x_k)$$

- ✧ If we consider that the linear model is exact then we pick $x_{k+1} = x^\star$ hence

$$0 = F(x_k) + J(x_k)(x_{k+1} - x_k)$$

At the end, we can add a step length $\alpha_k > 0$ in order to ensure a condition on the new iterate x_{k+1} or nuance the importance given to the search direction obtained by solving the linear system

$$J(x_k)d_k = -F(x_k) \tag{1.7}$$

For determined systems of nonlinear equations, if $J(x_k)$ is invertible then we can simply write one step of the method as

$$x_{k+1} = x_k - \alpha_k J(x_k)^{-1} F(x_k) \tag{1.8}$$

Nonetheless, in the general case, the uniqueness of the search direction d_k given by equation (1.7) is not guaranteed especially when $m > n$. Even worst, the linear system may not admit a solution so the direction is not defined. If we consider the Newton method where all the steps are undamped meaning $\alpha_k = 1$ for all k then we have the following theorem from Polyak [19] about the local rate of convergence for determined systems of equations (another result concerning the underdetermined case can also be found in the same reference).

Theorem 1.1. — *Suppose $F : \mathbb{R}^n \to \mathbb{R}^n$ is twice continuously differentiable on the set $B = \left\{x \in \mathbb{R}^n \;\middle|\; ||x - x_0|| \leq r\right\}$ such that $J(x_0)$ is invertible. Furthermore, we have*

$$||J(x_0)^{-1}F(x_0)|| \leq K_1$$
$$||J(x_0)^{-1}H(x)|| \leq K_2 \quad \forall x \in B$$

where K_1, K_2 are constants and $H(x)$ is the Hessian of $F(x)$. If the previous constants satisfy

$$h = K_1 K_2 < \frac{1}{2}, \quad r \geq \frac{1 - \sqrt{1 - 2h}}{h} K_1$$

then it exists a solution $x^\star \in B$ and the iterations (1.8) with undamped steps are always well-defined and converge to $x^\star$ with

$$||x_k - x^\star|| \leq \frac{K_1}{h 2^k} (2h)^{2^k}$$

From this theorem, we can deduce that the error committed at iteration k will reduce at least proportionally to q^{2^k} with $0 < q < 1$ which is quite fast. We say that the Newton method converges with a quadratic rate. In other words, the sequence $(x_k)_{k=0}^{\infty}$ converges and there exists a constant $c > 0$ and $k_0 \in \mathbb{N}$ such that

$$||x_{k+1} - x^\star|| \leq c \, ||x_k - x^\star||^2 \quad \text{for any } k \geq k_0$$

Although this rate of convergence is among the fastest that we can expect in practice, it has also few drawbacks that might make it computationally too expensive or even not applicable to certain classes of problems. On one hand, we need to solve a linear system of size n at each iteration which can be very expensive for large scale problems. On the other hand, the method also requires to compute all the entries of $J(x_k)$ and then store them in memory which is sometimes quite costly.

1.4 Quasi-Newton method

The quasi-Newton method is a derivative-free alternative to the Newton method where we try to estimate the Jacobian with an approximation $B_k \approx J(x_k)$.

Algorithm 2: Quasi-Newton method | Suitable for $n \geq m$

Data: Starting point $x_0 \in \mathbb{R}^n$
Initial approximation $B_0 \in \mathbb{R}^{m \times n}$

Step 0. Set $k = 0$

Step 1. Solve $B_k d_k = -F(x_k)$

Step 2. Pick a step size $\alpha_k > 0$

Step 3. $x_{k+1} = x_k + \alpha_k d_k$

Step 4. Build $B_{k+1} \approx J(x_{k+1})$

Step 5. Do $k \leftarrow k+1$ and go back to Step 1.

Again, if $m = n$ and B_k^{-1} exists then we have the update

$$x_{k+1} = x_k - \alpha_k B_k^{-1} F(x_k) \tag{1.9}$$

It is in general not possible to prove the quadratic convergence for the quasi-Newton method. Under reasonable assumptions, one can show that the method converges super-linearly namely

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

Notice that quadratic convergence also implies super-linear convergence. The super-linear convergence of the quasi-Newton method is stated formally in the next theorem, a complete proof can be found in Dennis and Moré [8].

Theorem 1.2. — *Suppose $F : \mathbb{R}^n \to \mathbb{R}^n$ is continuously differentiable on some open convex set $D \subset \mathbb{R}^n$ and there exists $x^\star \in D$ such that $F(x^\star) = 0$ and $J(x^\star)$ is invertible. Let $(x_k)_{k=0}^\infty$ be the sequence of iterates generated by recurrence (1.9) for $\alpha_k = 1$ where the matrices $(B_k)_{k=0}^\infty$ are all non-singular. If the sequence stays in D and $\lim_{k\to\infty} x_k = x^\star$ with $x_k \neq x^\star$ for any $k \geq 0$ then $(x_k)_{k=0}^\infty$ converges super-linearly if and only if*

$$\lim_{k \to \infty} \frac{||(B_k - J(x^\star))(x_{k+1} - x_k)||}{||x_{k+1} - x_k||} = 0 \tag{1.10}$$

Intuitively, the assumption (1.10) says that the approximation B_k should converge to the true Jacobian $J(x^\star)$ along the search direction. Usually, most of the quasi-Newton methods are defined such that the equality (1.11) called secant equation is satisfied.

$$B_{k+1} s_k = y_k \tag{1.11}$$

where $s_k = x_{k+1} - x_k$ and $y_k = F(x_{k+1}) - F(x_k)$. Thereafter, we give a sketch showing that when $(B_k)_{k=0}^\infty$ is a convergent sequence then (1.11) is a sufficient condition for (1.10) to hold, indeed

$$||(B_k - J(x^\star))(x_{k+1} - x_k)|| \leq ||(B_k - B_{k+1})s_k|| + ||(B_{k+1} - J(x^\star))s_k||$$

We deduce from the secant equation (1.11) and the Taylor expansion of $F(x_{k+1})$ that

$$\begin{aligned} ||(B_{k+1} - J(x^\star))s_k|| &= ||y_k - J(x^\star)s_k|| \\ &= ||F(x_{k+1}) - F(x_k) - J(x^\star)(x_{k+1} - x_k)|| \\ &= ||(J(\lambda x_{k+1} + (1-\lambda)x_k) - J(x^\star))(x_{k+1} - x_k)|| \end{aligned}$$

for some $0 < \lambda < 1$. Finally, since J is continuous and $x^\star = \lim_{k\to\infty} x_k$, we get

$$\begin{aligned} &\lim_{k\to\infty} \frac{||(B_k - J(x^\star))(x_{k+1} - x_k)||}{||x_{k+1} - x_k||} \\ &\leq \lim_{k\to\infty} ||B_{k+1} - B_k|| + ||(J(\lambda x_{k+1} + (1-\lambda)x_k) - J(x^\star))|| \\ &= 0 \end{aligned}$$

Moreover, the family of quasi-Newton methods in which the new matrix B_{k+1} is built by adding a symmetric low-rank perturbation to B_k are particularly efficient to deal with the matrix storage since we only need to store the perturbations. This class of methods can also avoid to solve the linear system at each iteration either by approximating directly the inverse of the Jacobian $H_k \approx J(x_k)^{-1}$ or via an explicit formula for B_k^{-1}, see for example Broyden [4], Dennis and Moré [8].

To sum up, the quasi-Newton method is in general much more cheaper in terms of computation by iteration than the Newton method only at the cost of losing the quadratic convergence. Therefore, this method is often considered as one of the most efficient method for solving large scale systems of nonlinear equations.

1.5 Spectral method

Firstly introduced in Barzilai and Borwein [2], the spectral method is a quasi-Newton method for solving determined systems of equations where we do the rough approximation $B_k = \frac{1}{\sigma_k} I$. The scalar σ_k is called spectral coefficient and must be such that $|\sigma_k|$ is bounded away from zero and not too large in other words

$$\sigma_{\min} \leq |\sigma_k| \leq \sigma_{\max}$$

for some constants $\sigma_{\min},\ \sigma_{\max} > 0$. This very simple approximation allows to compute directly $d_k = -\sigma_k F(x_k)$ instead of having to solve a linear system. The algorithm corresponding to the spectral method is presented below.

Algorithm 3: Spectral method | Suitable for $n = m$

Data: Starting point $x_0 \in \mathbb{R}^n$
Bounds $0 < \sigma_{\min} \leq \sigma_{\max} < \infty$
Initial spectral coefficient $\sigma_0 \in \mathbb{R}$ with $\sigma_{\min} \leq |\sigma_0| \leq \sigma_{\max}$

Step 0. Set $k = 0$

Step 1. $d_k = -\sigma_k F(x_k)$

Step 2. Pick a step size $\alpha_k > 0$

Step 3. $x_{k+1} = x_k + \alpha_k d_k$

Step 4.0. $\sigma = \frac{s_k^T s_k}{s_k^T y_k}$ with $s_k = x_{k+1} - x_k$ and $y_k = F(x_{k+1}) - F(x_k)$

Step 4.1. If $\sigma_{\min} \leq |\sigma| \leq \sigma_{\max}$ then set $\sigma_{k+1} = \sigma$

Step 4.2. Otherwise take arbitrary σ_{k+1}
such that $\sigma_{\min} \leq |\sigma_{k+1}| \leq \sigma_{\max}$

Step 5. Do $k \leftarrow k + 1$ and go back to Step 1.

The spectral method applied to the optimization problem $\min_x g(x)$, which can be done by looking for stationary points so $F(x) = \nabla g(x) = 0$, corresponds to a gradient method. We refer to Birgin, Martínez, and Raydan [3] for additional information.

Notice that $\sigma_{\max}$ and $\sigma_{\min}$ are often chosen very large and very small respectively, typically $\sigma_{\min} = 10^{-10}, \sigma_{\max} = 10^{10}$. It implies that the constraints on the spectral coefficient are almost technical assumptions hence for most of the iterations, we have

$$\sigma_{k+1} = \frac{s_k^T s_k}{s_k^T y_k} \tag{1.12}$$

The formula (1.12) is mainly motivated by the secant equation (1.11) which becomes for the spectral method

$$B_{k+1} s_k = \frac{1}{\sigma_{k+1}} s_k = y_k$$

Let's define $\lambda_{k+1} = \frac{1}{\sigma_{k+1}}$. Obviously the equation $\lambda_{k+1} s_k = y_k$ has nearly no chance to be true but if we consider the least square solution then we obtain

$$\begin{aligned} \lambda_{k+1} &= \arg\min_{\lambda} \ ||\lambda s_k - y_k||^2 \\ &= \arg\min_{\lambda} \ s_k^T s_k \lambda^2 - 2 s_k^T y_k \lambda + y_k^T y_k \\ &= \frac{s_k^T y_k}{s_k^T s_k} \qquad \Longrightarrow \quad \sigma_{k+1} = \frac{s_k^T s_k}{s_k^T y_k} \end{aligned}$$

If we solve instead the least square problem

$$\tilde{\sigma}_{k+1} = \min_{\sigma} ||\sigma y_k - s_k||^2$$

then we get a different spectral coefficient provided by the formula (1.13).

$$\tilde{\sigma}_{k+1} = \frac{s_k^T y_k}{y_k^T y_k} \tag{1.13}$$

One can check that the spectral method for $n = 1$ with the spectral coefficient given either by (1.12) or (1.13) reduces to the well-known secant method :

$$x_{k+1} = x_k - \alpha_k \frac{x_k - x_{k-1}}{F(x_k) - F(x_{k-1})} F(x_k)$$

where $J(x_k)$ is approximated via the finite difference

$$J(x_k) \approx \frac{F(x_k) - F(x_{k-1})}{x_k - x_{k-1}}$$

Nevertheless, the spectral coefficients (1.12) and (1.13) results generally in different methods which are highly nonmonotone and relatively efficient given the very low computational cost of an iteration.

1.6 Line search

Until now, we don't really specify a way to find "good" values for $\alpha_k > 0$. We might think about taking $\alpha_k = 1$ for all k. The iterations with the unit step are said undamped and lead usually to a faster convergence. Nevertheless, purely undamped methods are relatively hazardous for general problems as the steps might become out of control and generate iterates that never succeed to reach the solution.

Another approach consists to define beforehand a given sequence $(\alpha_k)_{k=0}^{\infty}$ describing the step lengths, for instance $\alpha_k = \frac{1}{k+1}$. The idea is that at the beginning we are enough confident to make large steps but as we get closer to the solution, we try to reduce the size of the steps in order to avoid missing the solution.

Concerning the methods presented here, we will mainly focus on a dynamical procedure called line search which computes α_k such that $x_{k+1} = x_k + \alpha_k d_k$ satisfies some condition. The line search algorithms are widely used in the optimization field and usually require the computation of $f(x_k + \alpha_k d_k)$. The name "line search" comes from the fact that we only look at f along the search direction d_k. Since we are looking for a minimum, it would make sense to take α_k such that the objective function f is minimized in the given direction, thus

$$\alpha_k = \arg\min_{\alpha > 0} f(x_k + \alpha d_k) \tag{1.14}$$

The strategy (1.14) is known as "exact line search". It works well for structured problems but in general solving a one dimensional optimization problem can be quite expensive. Hence, most of the line searches will try to find

$$\alpha_k \approx \arg\min_{\alpha > 0} f(x_k + \alpha d_k)$$

However, this choice may not be appropriate when the method which provides the search direction has a highly nonmonotone behavior. The line search condition corresponds in general to an inequality which can be written without loss of generality as $\phi(\alpha) \geq 0$ for some function $\phi : \mathbb{R}_+ \to \mathbb{R}$. If $\phi(\alpha) \geq 0$ then we said that α is an

admissible step size for the line search otherwise $\phi(\alpha) < 0$ and we need to test again the condition with another value.

The next algorithm try to find α such that $\phi(\alpha) \geq 0$ by testing values decreasing geometrically. We start with $\bar{\alpha}$ an initial guess (often $\bar{\alpha} = 1$ because the undamped step is more effective). Afterwards, we decrease our last value tested by a factor $\beta \in (0, 1)$ until the condition is satisfied.

Algorithm 4: General line search

Data: Rate of decrease $\beta \in (0, 1)$
Initial guess $\bar{\alpha} \in \mathbb{R}_+$

Step 0. Set $l = 0$

Step 1. $\alpha = \beta^l \bar{\alpha}$

Step 2. If $\phi(\alpha) \geq 0$ then return α

Step 3. Do $l \leftarrow l + 1$ and go back to Step 1.

Notice that if ϕ is a continuous function such that $\phi(0) > 0$ (meaning $\alpha = 0$ is admissible with the strict inequality) then the line search procedure above will terminate in a finite number of steps. Indeed, by definition of the continuity, ϕ can't change of sign instantly so there exists $\delta > 0$ such that for all $\alpha \in [0, \delta]$, we have $\phi(\alpha) \geq 0$. Hence, it exists an interval of small admissible step sizes that we will reach if we take α sufficiently small.

Ideally, we would like to have a line search specialized for nonmonotone and derivative-free methods so that we can combine it with the spectral method described previously. The basic line search, proposed by Armijo [1] is

$$f(x_k + \alpha d_k) \leq f(x_k) + c\alpha \nabla f(x_k)^T d_k \quad \text{with} \quad 0 < c < 1 \tag{1.15}$$

Inequality (1.15) requires the objective function at the next iterate to be smaller than the tangent to f in x_k, where we raise the slope by a factor c. The existence of sufficiently small admissible step sizes can be proven but only when $\nabla f(x_k)^T d_k < 0$. Because for $\alpha \to 0$, we have

$$\frac{f(x_k + \alpha d_k) - f(x_k)}{\alpha} = \nabla f(x_k)^T d_k < c \nabla f(x_k)^T d_k$$

An issue with the Armijo line search is that we always impose that the next iterate decreases the objective value. In order to allow a nonmonotone behavior, Grippo, Lampariello, and Lucidi [12] proposed the following line search

$$f(x_k + \alpha d_k) \leq \max_{0 \leq j \leq \min(k, M-1)} f(x_{k-j}) + c\alpha \nabla f(x_k)^T d_k \quad \text{with} \quad 0 < c < 1 \quad \text{and} \quad M \in \mathbb{N}_{>0} \tag{1.16}$$

This line search is a generalization of (1.15) since we take the maximum function value over the last M iterates instead of only the current one. Similar results hold for the existence of an admissible value of α as if (1.15) is satisfied then it will also be the case for (1.16). However, both line searches make use of $\nabla f(x_k)$ which is not

allowed in the context of derivative-free methods. In addition, the requirement of using only descent directions can't be done easily in our set up. For instance, if we consider the search direction provided by the spectral method $d_k = -\sigma_k F(x_k)$ then we can see that $\nabla f(x_k)^T d_k = -\sigma_k F(x_k)^T J(x_k) F(x_k) < 0$ can't be guaranteed easily.

A line search that fixes some of these issues is the one from Li and Fukushima [15] :

$$||F(x_k + \alpha d_k)|| \leq (1 + \theta_k)\,||F(x_k)|| - \rho\alpha^2 ||d_k||^2 \tag{1.17}$$
$$\text{with} \quad \rho > 0 \quad \text{and} \quad \theta_k > 0 \text{ such that } \sum_{k=0}^{\infty} \theta_k < \infty$$

Here, we can prove that the line search will terminate in finitely many steps without any restriction on the search direction because we can write (1.17) as

$$\phi(\alpha) = (1 + \theta_k)||F(x_k)|| - ||F(x_k + \alpha d_k)|| - \rho\alpha^2 ||d_k||^2 \geq 0$$

and ϕ is continuous with $\phi(0) = \theta_k ||F(x_k)|| > 0$ when x_k is not the solution. Contrary to (1.16), a method using the line search (1.17) will behave asymptotically as a monotone method because $\sum_{k=0}^{\infty} \theta_k < \infty$ which implies that $\theta_k \to 0$ quickly enough. Therefore, we have for large values of k

$$0 < \rho\alpha^2 ||d_k||^2 \leq (1 + \theta_k)||F(x_k)|| - ||F(x_k + \alpha d_k)|| \approx ||F(x_k)|| - ||F(x_k + \alpha d_k)||$$

Cruz, Martínez, and Raydan [6] describe a new line search combining features of both line searches (1.16) and (1.17) which is particularly well-adapted for nonmonotone derivative-free methods namely

$$f(x_k + \alpha d_k) \leq \max_{0 \leq j \leq \min(k, M-1)} f(x_{k-j}) + \theta_k - \rho\alpha^2 f(x_k) \tag{1.18}$$
$$\text{with} \quad \rho > 0, \quad M \in \mathbb{N}_{>0} \quad \text{and} \quad \theta_k > 0 \text{ such that } \sum_{k=0}^{\infty} \theta_k < \infty$$

The nonmonotone behavior of (1.18) is mostly ensured by

$$\max_{0 \leq j \leq \min(k, M-1)} f(x_{k-j})$$

and this effect becomes more important when we take large values for M. This nonmonotonicity may also come from $\theta_k > 0$, nevertheless as explained before this quantity tends relatively rapidly to zero as k increases. Finally, the purpose of the term $-\rho\alpha^2 f(x_k)$ is to obtain theoretical convergence results which will be explained in the next chapter. Again, we can check that line search procedures that decrease the step size at each iteration will terminate because

$$\phi(\alpha) = \max_{0 \leq j \leq \min(k, M-1)} f(x_{k-j}) - f(x_k + \alpha d_k) + \theta_k - \rho\alpha^2 f(x_k)$$

is continuous and satisfies

$$\phi(0) = \max_{0 \leq j \leq \min(k, M-1)} f(x_{k-j}) - f(x_k) + \theta_k \geq \theta_k > 0$$

1.7 DF-SANE

The method DF-SANE introduced by Cruz, Martínez, and Raydan [6] is at the origin of the algorithm presented in this thesis. The name DF-SANE stands for "**D**erivative-**F**ree **S**pectral **A**lgorithm for **N**on-linear **E**quations". It is a spectral method where we consider two possible search directions $d_k = \pm\sigma_k F(x_k)$ combined with the line search (1.18), as described in what follows :

Algorithm 5: DF-SANE | Suitable for $n = m$

Data: Starting point $x_0 \in \mathbb{R}^n$
Bounds $0 < \sigma_{\min} \leq \sigma_{\max} < \infty$
Sequence $(\theta_k)_{k=0}^{\infty}$ with $\theta_k > 0$ and $\sum_{k=0}^{\infty} \theta_k < \infty$
Line search parameters $\rho > 0,\ M \in \mathbb{N}_{>0},\ \beta \in (0,1)$

Step 0. Set $k = 0$

Step 1. Compute σ_k such that $\sigma_{\min} \leq |\sigma_k| \leq \sigma_{\max}$
Define $d_k = -\sigma_k F(x_k)$

Step 2.0. Set $l = 0$
$\bar{f}_k = \max_{0 \leq j \leq \min(k, M-1)} f(x_{k-j})$

Step 2.1. If $f(x_k + \beta^l d_k) \leq \bar{f}_k + \theta_k - \rho \left(\beta^l\right)^2 f(x_k)$
then set $\alpha_k = \beta^l,\ d = d_k$ and go to Step 3.

Step 2.2. If $f(x_k - \beta^l d_k) \leq \bar{f}_k + \theta_k - \rho \left(\beta^l\right)^2 f(x_k)$
then set $\alpha_k = \beta^l,\ d = -d_k$ and go to Step 3.

Step 2.3. Do $l \leftarrow l + 1$ and go back to Step 2.1

Step 3. $x_{k+1} = x_k + \alpha_k d$

Step 4. Do $k \leftarrow k + 1$ and go back to Step 1.

In the paper of Cruz, Martínez, and Raydan [6], the following theorem is proved about DF-SANE.

Theorem 1.3. — *Suppose $F : \mathbb{R}^n \to R^n$ is continuously differentiable and $(x_k)_{k=0}^{\infty}$ is a sequence generated by DF-SANE. If $\bar{x}$ is a limit point of the subsequence $(x_{v(j)-1})_{j=0}^{\infty}$ with*

$$v(j) = \arg \max_{(j-1)M+1 \leq k \leq jM} f(x_k)$$

then

$$F(\bar{x})^T J(\bar{x}) F(\bar{x}) = 0 \tag{1.19}$$

This result remains valid when we take the merit function $f(x) = ||F(x)||$ but for $f(x) = \frac{1}{2}||F(x)||^2$, we can write (1.19) equivalently as

$$\nabla f(\bar{x})^T F(\bar{x}) = 0$$

hence at any limit point $\bar{x}$ the gradient ∇f and the residual F are orthogonal. This means that we might have $F(\bar{x}) \neq 0$. However, if the mapping F is μ-strongly monotone, then

$$0 = F(\bar{x})^T J(\bar{x}) F(\bar{x}) \geq \mu ||F(\bar{x})||^2 \quad \Longrightarrow \quad F(\bar{x}) = 0$$

Finally, note that none of the theoretical results requires explicitly $\sigma_{k+1} = \sigma$ with

$$\sigma = \frac{s_k^T s_k}{s_k^T y_k} \qquad \text{if} \quad \sigma_{\min} \leq |\sigma| \leq \sigma_{\max}$$

but the implementation is usually done in the same way as with the spectral method because it works well in practice.

1.8 DF-SAUNE

For underdetermined systems, DF-SANE can't be applied directly. Indeed, one can simply remark that it is not possible to sum up the search direction $d_k = -\sigma_k F(x_k) \in \mathbb{R}^m$ with the iterate $x_k \in \mathbb{R}^n$. The paper of Echebest, Schuverdt, and Vignau [9] gives a method equivalent to DF-SANE from Cruz, Martínez, and Raydan [6] when $n = m$ but with the possibility to handle also the case $n > m$. Analogously to DF-SANE, the author called this new method DF-SAUNE for "**D**erivative-**F**ree **S**pectral **A**lgorithm for **U**nderdetermined **N**on-linear **E**quations". DF-SAUNE is inspired by the augmented Jacobian algorithm presented in Walker and Watson [21]. In this method, the search direction $d_k \in \mathbb{R}^n$ is found by solving the square system of linear equations

$$\begin{pmatrix} J(x_k) \\ V_k \end{pmatrix} d_k = \begin{pmatrix} -F(x_k) \\ 0 \end{pmatrix} \tag{1.20}$$

where $V_k \in \mathbb{R}^{(n-m)\times n}$ is chosen such that the full matrix in equation (1.20) is invertible. As a reminder for the spectral method with $m = n$, we approximate the Jacobian by some multiple of the identity matrix nevertheless here the Jacobian is not square hence we take instead

$$J(x_k) \approx \frac{1}{\sigma_k} E_k \tag{1.21}$$

where $E_k \in \mathbb{R}^{m\times n}$ is a matrix with a very simple structure which reduces to $E_k = I$ for all k in the determined case. The authors Echebest, Schuverdt, and Vignau [9] choose a periodic sequence $(E_k)_{k=0}^{\infty}$ meaning that we repeat infinitely a sequence $E_0, E_1, \ldots, E_{N-1}$ with $N = \left\lceil \frac{n}{m} \right\rceil$. If n is divisible by m then the matrix $E_0, E_1, \ldots, E_{N-1}$ are made of N blocks with size $m \times m$ as follow

$$E_j = \begin{pmatrix} 0 & 0 & \cdots & I & \cdots & 0 \end{pmatrix} \tag{1.22}$$

where the jth block that composes E_j is the identity matrix and the others are set to zero. When $\frac{n}{m}$ is not an integer $E_0, E_1, \ldots, E_{N-2}$ are built in the same way but the last block of E_{N-1} will only consists in the $n - (N-1)m$ first columns of the identity matrix and the remaining columns will be the first columns of E_{N-1}. We clarify the above with a simple example for $m = 3$ and $n = 7$ so $N = \left\lceil \frac{n}{m} \right\rceil = 3$

$$E_0 = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{pmatrix}$$

$$E_1 = \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

$$E_2 = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Combining the ideas of equations (1.20) and (1.21), we get

$$\frac{1}{\sigma_k}\begin{pmatrix} E_k \\ V_k \end{pmatrix} d_k = \begin{pmatrix} -F(x_k) \\ 0 \end{pmatrix} \tag{1.23}$$

If we choose V_k such that the rows of E_k and V_k form the canonical basis of $\mathbb{R}^n$ then $\begin{pmatrix} E_k \\ V_k \end{pmatrix}$ is invertible and also orthogonal hence multiplying equation (1.23) by the transpose gives

$$\begin{aligned} \frac{1}{\sigma_k} d_k = \begin{pmatrix} E_k^T & V_k^T \end{pmatrix} \begin{pmatrix} -F(x_k) \\ 0 \end{pmatrix} = -E_k^T F(x_k) \\ \implies \quad d_k = -\sigma_k E_k^T F(x_k) \end{aligned}$$

This particular choice of search direction leads to DF-SAUNE, we give thereafter the pseudo-code for completeness but there are only few modifications compared to DF-SANE. Notice that we can write

$$\begin{aligned} f(x_{k+1}) &\leq \max_{0 \leq j \leq \min(k, M-1)} f(x_{k-j}) + \theta_k - \rho \alpha^2 ||d_k||^2 \\ &= \max_{0 \leq j \leq \min(k, M-1)} f(x_{k-j}) + \theta_k - 2\rho \alpha^2 \sigma_k^2 f(x_k) \end{aligned}$$

because

$$||d_k||^2 = \sigma_k^2 ||E_k^T F(x_k)||^2 = \sigma_k^2 ||F(x_k)||^2 = 2\sigma_k^2 f(x_k)$$

thus the line searches of DF-SANE and DF-SAUNE are also closely related.

Algorithm 6: DF-SAUNE | Suitable for $n \geq m$

Data: Starting point $x_0 \in \mathbb{R}^n$
Bounds $0 < \sigma_{\min} \leq \sigma_{\max} < \infty$
Sequence $(\theta_k)_{k=0}^{\infty}$ with $\theta_k > 0$ and $\sum_{k=0}^{\infty} \theta_k < \infty$
Line search parameters $\rho > 0,\ M \in \mathbb{N}_{>0},\ \beta \in (0,1)$
Matrices $(E_k)_{k=0}^{\infty}$ with size $m \times n$, see definition (1.22)

Step 0. Set $k = 0$

Step 1. Compute σ_k such that $\sigma_{\min} \leq |\sigma_k| \leq \sigma_{\max}$
$d_k = -\sigma_k E_k^T F(x_k)$

Step 2.0. Set $l = 0$
$\bar{f}_k = \max_{0 \leq j \leq \min(k, M-1)} f(x_{k-j})$

Step 2.1. If $f(x_k + \beta^l d_k) \leq \bar{f}_k + \theta_k - \rho \left(\beta^l\right)^2 ||d_k||^2$
then set $\alpha_k = \beta^l,\ d = d_k$ and go to Step 3.

Step 2.2. If $f(x_k - \beta^l d_k) \leq \bar{f}_k + \theta_k - \rho \left(\beta^l\right)^2 ||d_k||^2$
then set $\alpha_k = \beta^l,\ d = -d_k$ and go to Step 3.

Step 2.3. Do $l \leftarrow l + 1$ and go back to Step 2.1

Step 3. $x_{k+1} = x_k + \alpha_k d$

Step 4. Do $k \leftarrow k + 1$ and go back to Step 1.

An adaptation of the proof in Cruz, Martínez, and Raydan [6] for DF-SAUNE gives an almost identical result :

Theorem 1.4. — *Suppose $F : \mathbb{R}^n \to R^m$ is continuously differentiable and $(x_k)_{k=0}^{\infty}$ is a sequence generated by DF-SAUNE. If $\bar{x}$ is a limit point of the subsequence $(x_{v(j)-1})_{j=0}^{\infty}$ with*

$$v(j) = \arg \max_{(j-1)M+1 \leq k \leq jM} f(x_k)$$

then there exists $j_0 \in \{0, 1, \ldots, N-1\}$ such that

$$F(\bar{x})^T J(\bar{x}) E_{j_0}^T F(\bar{x}) = 0$$

1.9 General class of methods

In view of the astonishing numerical results of DF-SANE, see Cruz, Martínez, and Raydan [6], a more quantitative approach was needed in order to better understand the method theoretically. The paper from Grapiglia and Chorobura [11] presented a general class of derivative-free nonmonotone methods which includes DF-SANE as a particular case. The line search condition is extended as follow

$$f(x_{k+1}) \leq f(x_k) + \nu_k + \theta_k - \rho \alpha^2 f(x_k) \tag{1.24}$$

$$\text{with} \quad \rho > 0, \quad \nu_k \geq 0 \quad \text{and} \quad \theta_k > 0 \text{ such that } \sum_{k=0}^{\infty} \theta_k \leq \theta < \infty$$

We can show that (1.24) leads to a well-defined line search procedure with a similar reasoning as for (1.18) because $\nu_k + \theta_k > 0$. The nonmonotone behavior of this line search is mainly characterised by the choices for the sequence $(\nu_k)_{k=0}^{\infty}$ which is not necessarily a summable sequence. If we decide to define

$$\nu_k = -f(x_k) + \max_{0 \leq j \leq \min(k, M-1)} f(x_{k-j})$$

then we recover the line search of DF-SANE.

Algorithm 7: General method | Suitable for $n = m$

Data: Starting point $x_0 \in \mathbb{R}^n$
Bounds $0 < \sigma_{\min} \leq \sigma_{\max} < \infty$
Sequence $(\nu_k)_{k=0}^{\infty}$ with $\nu_k \geq 0$
Sequence $(\theta_k)_{k=0}^{\infty}$ with $\theta_k > 0$ and $\sum_{k=0}^{\infty} \theta_k \leq \theta < \infty$
Line search parameters $\rho > 0,\ \beta \in (0,1)$

Step 0. Set $k = 0$

Step 1. Compute σ_k such that $\sigma_{\min} \leq |\sigma_k| \leq \sigma_{\max}$
$d_k = -\sigma_k F(x_k)$

Step 2.0. Set $l = 0$

Step 2.1. If $f(x_k + \beta^l d_k) \leq f(x_k) + \nu_k + \theta_k - \rho \left(\beta^l\right)^2 f(x_k)$
then set $l_k = l,\ \alpha_k = \beta^{l_k},\ d = d_k$ and go to Step 3.

Step 2.2. If $f(x_k - \beta^l d_k) \leq f(x_k) + \nu_k + \theta_k - \rho \left(\beta^l\right)^2 f(x_k)$
then set $l_k = l,\ \alpha_k = \beta^{l_k},\ d = -d_k$ and go to Step 3.

Step 2.3. Do $l \leftarrow l + 1$ and go back to Step 2.1

Step 3. $x_{k+1} = x_k + \alpha_k d$

Step 4. Do $k \leftarrow k + 1$ and go back to Step 1.

The authors focus on finding a point $\bar{x}$ such that $\psi(\bar{x}) \leq \epsilon$ where

$$\psi(x) = \begin{cases} \min\left(||F(x)||, \frac{|\nabla f(x)^T F(x)|}{||F(x)||}\right) & \text{if } F(x) \neq 0 \\ 0 & \text{otherwise} \end{cases} \tag{1.25}$$

The convergence analysis of the method is done under some hypothesis about the problem and the parameters of the method which are

H1 The Jacobian $J(x)$ is Lipschitz continuous.

H2 The sequence $(\nu_k)_{k=0}^{\infty}$ is summable so $\sum_{k=0}^{\infty} \nu_k \leq \nu < \infty$.

H3 The sublevel set $\mathcal{L}_f(x_0) = \{x \in \mathbb{R}^n \mid f(x) \leq f(x_0) + \nu + \theta\}$ is bounded.

We now give an important result from Grapiglia and Chorobura [11]

Theorem 1.5. — *Suppose that **H1**, **H2** and **H3** hold and let $(x)_{k=0}^{\infty}$ be a sequence generated by Algorithm 8. Given a tolerance $\epsilon > 0$, we consider the set given by*

$$\Omega(\epsilon) = \{k \in \mathbb{N} \mid \psi(x_k) > \epsilon\}$$

then we can bound the number of elements in this set as follow

$$|\Omega(\epsilon)| \leq 2 \frac{f(x_0) + \nu + \theta}{\rho \min\left(1, \left(\frac{2\beta\sigma_{\min}}{\rho + L\sigma_{\max}^2}\right)^2\right)} \epsilon^{-2}$$

This theorem implies that the method is able to find $\bar{x}$ with $\psi(\bar{x}) \leq \epsilon$ within $\mathcal{O}\left(\epsilon^{-2}\right)$ iterations. Furthermore, it was also shown that if $\psi(x_k) > \epsilon$ for some $\epsilon \in (0, 1)$ then the step size at iteration k is bounded from above namely

$$\alpha_k \geq \tau\epsilon$$

for some constant $\tau > 0$. From there, we can deduce a bound for the number of function evaluations because we have

$$\alpha_k = \beta^{l_k} \geq \tau\epsilon \quad \Longrightarrow \quad l_k \leq \frac{\log(\tau\epsilon)}{\log\beta}$$

Notice that the number of function evaluations at iteration k is at most $2(l_k + 1)$ because for $l = 0, 1, \ldots l_k$ we need to check both directions (or potentially only one at the last step). Hence *Algorithm 7* with a summable sequence $(\nu_k)_{k=0}^{\infty}$ must perform $\mathcal{O}\left(\epsilon^{-2}\log(\epsilon^{-1})\right)$ computations of $F(x)$ to find $\bar{x}$ such that $\psi(\bar{x}) \leq \epsilon$.

Another method called N-DF-SANE from Cheng and Li [5] is also described and it is shown in Grapiglia and Chorobura [11], that this method belongs to the class of methods corresponding to *Algorithm 7*. N-DF-SANE is a variant of DF-SANE which demonstrates better results on most of the numerical tests compared to DF-SANE. The line search condition used in N-DF-SANE is

$$f(x_{k+1}) \leq C_k + \theta_k - \rho\alpha_k^2 f(x_k)$$
$$\text{with} \quad \rho > 0, \quad M \in \mathbb{N}_{>0}, \quad \text{and} \quad \theta_k > 0 \text{ such that } \sum_{k=0}^{\infty} \theta_k \leq \theta < \infty$$

The sequence $(C_k)_{k=0}^{\infty}$ characterizing the nonmonocity of the method is defined by

$$C_{k+1} = (1 - \delta_{k+1})(C_k + \theta_k) + \delta_{k+1} f(x_{k+1})$$
$$\text{with} \quad \delta_{k+1} \in (0, 1] \quad \text{and} \quad C_0 = f(x_0)$$

Afterwards, the problems with a μ-strongly monotone mapping are considered in Grapiglia and Chorobura [11] where we can remark that

$$\frac{|\nabla f(x)^T F(x)|}{||F(x)||} = \frac{|F(x)^T J(x) F(x)|}{||F(x)||} \geq \mu ||F(x)||$$

Therefore,

$$\psi(x) \leq \epsilon \quad \Longrightarrow \quad \min(1, \mu)\, ||F(x)|| \leq \epsilon \tag{1.26}$$

On top of this improvement in the previous results, they also consider two other methods using the following parameters

$$\nu_k = 0, \quad \theta_k = \gamma^k \theta_0 \quad \text{with } 0 < \gamma < 1 \text{ and } \theta_0 > 0$$

and prove complexity bounds in $\mathcal{O}\left(\log(\epsilon^{-1})\right)$ for both iterations and function evaluations to find an iterate $\bar{x}$ such that $f(\bar{x}) = \frac{1}{2}||F(\bar{x})||^2 \leq \epsilon$. Notice that according to (1.26), the general method *Algorithm 7* has only a complexity of $\mathcal{O}\left(\epsilon^{-1}\log(\epsilon^{-\frac{1}{2}})\right)$ to find $\bar{x}$ such that $\psi(\bar{x}) \leq \epsilon$. In particular, one of these two methods uses an adaptive approach for the step size that is to say

$$\alpha_{k+1} = \beta^{l_k - 1}\alpha_k$$
$$x_{k+1} = x_k + \alpha_k \beta^{l_k} d_k$$

Here, the notation α_k doesn't correspond to the step length anymore which is actually $\alpha_k\beta^{l_k}$. Instead, it is the guess that we make at the first iteration in the line search. This strategy keeps in memory information from the last step and we will investigate it further in Chapter 2.

1.10 Contributions

The goals of this work are :

- ✧ Present a new class of derivative-free methods
- ✧ Prove worst-case evaluation complexity for two new methods
- ✧ Experiment and compare numerically these new methods
- ✧ Provide a new way to generate adversarial example in the black-box setting

Chapter 2

New class of methods

In this chapter, we present a new class of methods for solving underdetemined systems of nonlinear equations. Then, we focus on two particular methods belonging to this class and prove complexity bounds regarding the number of function evaluations.

2.1 Search direction

The new class of derivative-free nonmonotone methods for solving underdetermined systems of nonlinear equations proposed in this thesis is a quasi-Newton method. More precisely, in such method, the search direction $d_k \in \mathbb{R}^n$ is computed by solving

$$B_k d_k = -F(x_k) \tag{2.1}$$

where $B_k \in \mathbb{R}^{m\times n}$ aims to approximate the Jacobian $J(x_k)$ and thus imitates the "ideal" behavior of the Newton method. We have seen in *Chapter 1* that methods choosing a simple structure for B_k in order to be very cheap, remain quite competitive in practice. We consider the approximation of the Jacobian given by

$$B_k = \frac{1}{\sigma_k} E_k \approx J(x_k) \tag{2.2}$$

We remind that $\sigma_k \in \mathbb{R}$ is the spectral coefficient which must be bounded as follow

$$0 < \sigma_{\min} \leq |\sigma_k| \leq \sigma_{\max} < \infty$$

with some constant bounds $\sigma_{\min}$ and $\sigma_{\max}$. Nevertheless, it is almost a technical assumption required in the proofs as we often pick $\sigma_{\min}$, $\sigma_{\max}$ respectively small and large enough. The matrix E_k in equation (2.2) somehow represents approximately the structure of the Jacobian. As a reminder, the spectral method presented in *Chapter 1* is such that $E_k = I$ for all k hence we would like to generalize it to underdetermined systems of equations.

In the new method, we define $E_k \in \mathbb{R}^{m\times n}$ as a matrix whose rows are distinct and belongs to the set $\{e_1, e_2, \ldots, e_n\}$. The vector $e_i \in \mathbb{R}^n$ is made of zeros except the ith component which is equal to 1. One can easily check that if $m = n$ then we recover the spectral direction.

An important property which will be used again later is that for any matrix E_k defined according to the previous rule, we have

$$\boxed{E_k E_k^T = I} \tag{2.3}$$

Notice that all the proofs done in this chapter are still valid if we only require that each matrix in the sequence $(E_k)_{k=0}^{\infty}$ satisfies the equation (2.3). However, using matrices with entries in $\{0,1\}$ allows to compute very quickly the matrix-vector product. Moreover, a valid search direction in the sense of (2.1) is provided by

$$\boxed{d_k = -\sigma_k E_k^T F(x_k)} \tag{2.4}$$

Indeed, if we use equation (2.3) then we find

$$B_k d_k = \left(\frac{1}{\sigma_k} E_k\right)\left(-\sigma_k E_k^T F(x_k)\right) = -E_k E_k^T F(x_k) = -F(x_k)$$

Thus, the definition of B_k can be interpreted as follow :

Choose m variables and consider the other $n-m$ as constants. Afterwards the new system of equations has m unknowns and m equations so we can apply a spectral method step to the corresponding subset of variables.

The formula for the spectral coefficient used in practice when its absolute value is contained in $[\sigma_{\min}, \sigma_{\max}]$ must be somewhat modified compared to the spectral method of *Chapter 1*. Again, we consider the secant condition

$$B_{k+1} s_k = \frac{1}{\sigma_{k+1}} E_{k+1} s_k = y_k \tag{2.5}$$

where $s_k = x_{k+1} - x_k$ and $y_k = F(x_{k+1}) - F(x_k)$. Given our choice of approximation for the Jacobian, it is quite unlikely that the equation (2.5) holds. Nevertheless, if we define $\lambda_{k+1} = \frac{1}{\sigma_{k+1}}$ and consider the least square solution

$$\lambda_{k+1} = \arg\min_{\lambda} \; ||E_{k+1} s_k \, \lambda - y_k||$$

then we get, thanks to the normal equations

$$(E_{k+1} s_k)^T (E_{k+1} s_k) \; \lambda_{k+1} = (E_{k+1} s_k)^T y_k$$

Hence, we deduce the formula

$$\sigma_{k+1} = \frac{(E_{k+1} s_k)^T (E_{k+1} s_k)}{(E_{k+1} s_k)^T y_k} \tag{2.6}$$

2.2 Line search

We remind that we aim to minimize the merit function

$$\boxed{f(x) = \frac{1}{2} \, ||F(x)||^2}$$

At each step, we consider either the search directions d_k provided by equation (2.4) or $-d_k$. We check the opposite direction of d_k too because it is not clear which one is suitable for the minimization of the merit function $f(x)$.

The line search is almost the same as Grapiglia and Chorobura [11] and corresponds to finding a step length $\alpha > 0$ such that we have for either

$$x_{k+1} = x_k + \alpha d_k \quad \text{or} \quad x_{k+1} = x_k - \alpha d_k$$

The next inequality which holds

$$\boxed{\begin{array}{c} f(x_{k+1}) \leq f(x_k) + \nu_k + \theta_k - \rho\gamma_k\alpha^2 f(x_k) \\ \text{with} \quad \rho > 0, \quad \nu_k \geq 0 \quad \text{and} \quad \theta_k > 0 \text{ such that } \sum_{k=0}^{\infty} \theta_k \leq \theta < \infty \end{array}} \tag{2.7}$$

The purpose of the parameter γ_k in inequality (2.7) is to be able to prove theoretical results for the line searches in both methods proposed in Grapiglia and Chorobura [11] and Echebest, Schuverdt, and Vignau [9] corresponding respectively to $\gamma_k = 1$ and $\gamma_k = 2\sigma_k^2$. Since the spectral coefficient σ_k is bounded, we have also

$$\gamma_{\min} \leq \gamma_k \leq \gamma_{\max} \tag{2.8}$$

where

$$\begin{array}{c} \gamma_{\max} = \gamma_{\min} = 1 \quad \text{for Grapiglia and Chorobura [11]} \\ \gamma_{\max} = 2\sigma_{\max}^2,\ \gamma_{\min} = 2\sigma_{\min}^2 \quad \text{for Echebest, Schuverdt, and Vignau [9]} \end{array}$$

As seen in *Chapter 1*, the summable sequence of positive numbers $(\theta_k)_{k=0}^{\infty}$ allows to have a well-defined line search and the second sequence $(\nu_k)_{k=0}^{\infty}$ can be used to design a large range of nonmonontone methods. Notice that the proofs thereafter will require that $\sum_{k=0}^{\infty} \nu_k < \infty$ but it is not necessarily the case for the general class of methods considered.

2.3 Different strategies

The general class of methods obtained with the search direction and the line search presented above is provided explicitly in *Algorithm 8*.

Algorithm 8: New class of methods | Suitable for $n \geq m$

Data: Starting point $x_0 \in \mathbb{R}^n$
Initial guess $\alpha_0 > 0$
Bounds $0 < \sigma_{\min} \leq \sigma_{\max} < \infty$
Sequence $(\nu_k)_{k=0}^{\infty}$ with $\nu_k \geq 0$
Sequence $(\theta_k)_{k=0}^{\infty}$ with $\theta_k > 0$ and $\sum_{k=0}^{\infty} \theta_k \leq \theta < \infty$
Line search parameters $\rho > 0$, $\beta \in (0,1)$ and γ_k defined as 1 or $2\sigma_k^2$
Matrices $(E_k)_{k=0}^{\infty}$ with size $m \times n$ and distinct rows in $\{e_1, e_2, \ldots, e_n\}$

Step 0. Set $k = 0$

Step 1. Compute σ_k such that $\sigma_{\min} \leq |\sigma_k| \leq \sigma_{\max}$
$d_k = -\sigma_k E_k^T F(x_k)$

Step 2.0. Set $l = 0$

Step 2.1. If $f(x_k + \alpha_k \beta^l d_k) \leq f(x_k) + \nu_k + \theta_k - \rho\gamma_k(\alpha_k\beta^l)^2 f(x_k)$
then set $l_k = l$, $d = d_k$ and go to Step 3.

Step 2.2. If $f(x_k - \alpha_k \beta^l d_k) \leq f(x_k) + \nu_k + \theta_k - \rho\gamma_k(\alpha_k\beta^l)^2 f(x_k)$
then set $l_k = l$, $d = -d_k$ and go to Step 3.

Step 2.3. Do $l \leftarrow l + 1$ and go back to Step 2.1

Step 3. $x_{k+1} = x_k + \alpha_k \beta^{l_k} d$
Pick initial guess α_{k+1} for the next iteration

Step 4. Do $k \leftarrow k + 1$ and go back to Step 1.

Notice that we have many degrees of freedom in *Algorithm 8* as the definition of the sequences $(\sigma_k)_{k=0}^{\infty}$, $(\nu_k)_{k=0}^{\infty}$, $(\theta_k)_{k=0}^{\infty}$, $(E_k)_{k=0}^{\infty}$ and $(\alpha_k)_{k=0}^{\infty}$ are not specified. Typically, σ_k is defined according to the formula (2.6) and we often set

$$\theta_k = \frac{\theta_0}{(k+1)^2} \quad \text{or} \quad \theta_k = \frac{\theta_0}{2^k}$$

with θ_0 a positive constant, one can check that we have $\sum_{k=0}^{\infty} \theta_k < \infty$.

Furthermore, if we define

$$\nu_k = -f(x_k) + \max_{0 \leq j \leq \min(k, M-1)} f(x_{k-j})$$

then we recover DF-SAUNE or it is also possible to simply pick $\nu_k = 0$.

Each matrix E_k translates mathematically which subset of m variables among n are updated at iteration k. We describe three possible ways of doing this without explicitly giving the corresponding matrices. The first two strategies use periodic sequence. It means that we repeat indefinitely the matrices $E_0, E_1, \ldots, E_{N-1}$ with $N \in \mathbb{N}$, the period of the sequence.

- ✧ **Standard strategy :** The sequence $(E_k)_{k=0}^{\infty}$ is periodic with $N = \left\lceil \frac{n}{m} \right\rceil$. It is the same strategy as DF-SAUNE from Echebest, Schuverdt, and Vignau [9], see also *Chapter 1*. Each matrix E_j for $j = 0, 1, \ldots, N-2$ defines the subset of

variables with indices

$$jm,\ jm+1,\ \ldots,\ (j+1)m-1$$

If n can be divided by m then the subset corresponding to E_{N-1} is defined similarly, otherwise we complete the last subset of variables with the first variables in order to have m variables in this subset.

- ✧ **Modulo strategy :** The sequence $(E_k)_{k=0}^{\infty}$ is periodic with N equal to the lowest common multiple of n and m. Each matrix E_j for $j = 0, 1, \ldots, N-1$ defines the subset of variables with indices

$$jm,\ jm+1,\ \ldots,\ (j+1)m-1 \qquad \text{modulo}(n)$$

where we use the modulo operator to say that when indices overcome n then we take the remainder of their division by n. Notice that when n is divisible by m then it is the same strategy as previously.

- ✧ **Random strategy :** Each matrix E_j for $j = 0, 1, \ldots, N-1$ defines a subset of m variables chosen randomly among n. This sequence is particularly useful when we don't want to have a regular pattern, see application in *Chapter 4*.

In the line search of *Algorithm 8*, we check successively for $l = 0, 1, 2, \ldots$ if the line search condition (2.7) is satisfied by the step size $\alpha = \alpha_k \beta^l$ where α_k is a guess value used at the first line search iteration. We present two different ways for defining the sequence of guess $(\alpha_k)_{k=0}^{\infty}$.

- ✧ **Memoryless method :** The first step size tested in the line search will always be the same so $\alpha_k = \alpha_0$ for all k. We call this approach "memoryless" because it doesn't take into account the information from the past steps. However, it is not really a drawback as we usually fix $\alpha_0 = 1$ hence we try first the undamped step which is often the most efficient.

- ✧ **Adaptive method :** This approach is called "adaptive" because the guess at iteration $k+1$ is adapted based on what we know from iteration k. More precisely, we pick $\alpha_{k+1} = \alpha_k \beta^{l_k - 1}$ meaning that the next guess is the effective step size at the previous iteration which is increased by a factor β^{-1}. Therefore, we do a guess which aims to be closer to the actual true step size in the hope of observing less function evaluations in the line search.

We will provide theoretical results for both Memoryless method and Adpative method regardless of the particular definition of the sequence $(E_k)_{k=0}^{\infty}$ but we will need an additional assumption about the sequence $(\nu_k)_{k=0}^{\infty}$.

2.4 Auxiliary results

In this section, we present four lemmas which will be important tools for carrying the convergence analysis of *Algorithm 8*. We consider the following assumptions :

A1 The Jacobian $J(x)$ is Lipschitz continuous with constant L_J

A2 The sequence $(\nu_k)_{k=0}^{\infty}$ is summable so $\sum_{k=0}^{\infty} \nu_k \leq \nu < \infty$

A3 The sublevel set $\mathcal{L}_f(x_0) = \{x \in \mathbb{R}^n \mid f(x) \leq f(x_0) + \nu + \theta\}$ is bounded namely $D_0 = \sup\{||x - x_0|| \mid x \in \mathcal{L}_f(x_0)\} < \infty$

Lemma 2.1. — *Any sequence $(x_k)_{k=0}^{\infty}$ generated by Algorithm 8 remains in the sublevel set*

$$\mathcal{L}_f(x_0) = \{x \in \mathbb{R}^n \mid f(x) \leq f(x_0) + \nu + \theta\}$$

Proof. By the line search condition (2.7), we have

$$f(x_{k+1}) \leq f(x_k) + \nu_k + \theta_k$$

Therefore

$$f(x_k) \leq f(x_0) + \sum_{j=0}^{k-1} (\nu_k + \theta_k) \leq f(x_0) + \nu + \theta$$

□

Lemma 2.2 and *Lemma 2.3* are mainly consequences of the fact that $J(x)$ is Lipschitz continuous. We use $\text{conv}(A)$ to denote the convex hull of $A \subset \mathbb{R}^n$.

Lemma 2.2. — *Under the assumptions **A1**, **A2** and **A3**, we have for all $x, y \in \text{conv}(\mathcal{L}_f(x_0))$*

$$||F(x) - F(y)|| \leq \left(L_J D_0 + ||J(x_0)||\right) ||x - y||$$

Proof. For any $x, y \in \text{conv}(\mathcal{L}_f(x_0))$, Taylor's expansion gives for some $\lambda \in [0, 1]$

$$F(x) = F(y) + J(\lambda x + (1 - \lambda)y)(x - y)$$

Therefore

$$\begin{aligned} ||F(x) - F(y)|| &\leq ||J(\lambda x + (1 - \lambda)y))||\, ||x - y|| \\ &\leq \left(||J(\lambda x + (1 - \lambda)y) - J(x_0)|| + ||J(x_0)|| \right) ||x - y|| \\ &\leq \left(L_J ||\lambda x + (1 - \lambda)y - x_0|| + ||J(x_0)|| \right) ||x - y|| \\ &\leq \left(L_J D_0 + ||J(x_0)|| \right) ||x - y|| \end{aligned} \tag{2.9}$$

The last inequality (2.9) comes from the fact that if $z \in \text{conv}(\mathcal{L}_f(x_0))$ then we have

$$||z - x_0|| \leq D_0$$

It can be shown by writing z as a convex combination of two elements in $\mathcal{L}_f(x_0)$.

□

Lemma 2.3. — *Under the assumptions **A1**, **A2** and **A3**, the function $f(x) = \frac{1}{2}||F(x)||^2$ restricted to the domain $\text{conv}(\mathcal{L}_f(x_0))$ has a L-Lipschitz gradient with*

$$L = \left(L_J D_0 + ||J(x_0)||\right)^2 + L_J D_0 \left(L_J D_0 + ||J(x_0)||\right) + L_J ||F(x_0)||$$

Proof. For any $x, y \in \text{conv}(\mathcal{L}_f(x_0))$

$$\begin{aligned} ||\nabla f(x) - \nabla f(y)|| &= ||J(x)^T F(x) - J(x)^T F(y) + J(x)^T F(y) - J(y)^T F(y)|| \\ &\leq ||J(x)||\, ||F(x) - F(y)|| + ||J(x) - J(y)||\, ||F(y)|| \end{aligned} \tag{2.10}$$

If we use *Lemma 2.2* and the Lipschitz continuity of $J(x)$ then equation (2.10) becomes

$$\begin{aligned}||\nabla f(x) - \nabla f(y)|| &\leq \left(||J(x) - J(x_0)|| + ||J(x_0)||\right)\left(L_J D_0 + ||J(x_0)||\right)||x-y|| \\ &\quad + L_J||x-y||\left(||F(y) - F(x_0)|| + ||F(x_0)||\right) \\ &\leq \Big(\left(L_J D_0 + ||J(x_0)||\right)^2 + L_J D_0(L_J D_0 + ||J(x_0)||) + L_J||F(x_0)||\Big)||x-y||\end{aligned}$$

□

In the following, when we assume that $J(x)$ is L_J-Lipschitz, the Lipschitz constant of $\nabla f(x)$ induced by *Lemma 2.3* will always be denoted by L. *Lemma 2.4* states a sufficient condition in order to have an admissible step size for the line search (2.7).

Lemma 2.4. — *Under the assumptions **A1**, **A2** and **A3**, let x_k be an iterate generated by Algorithm 8 such that $\nabla f(x_k)^T d_k \neq 0$ and*

$$0 < \alpha \leq c\,\frac{|(E_k \nabla f(x_k))^T F(x_k)|}{||F(x_k)||^2} \quad \textit{where} \quad c = \frac{2\sigma_{\min}}{L\sigma_{\max}^2 + \rho\gamma_{\max}}$$

then at least one of the two inequalities below is satisfied

$$\begin{aligned}f(x_k + \alpha d_k) &\leq f(x_k) + \nu_k + \theta_k - \rho\gamma_k\alpha^2 f(x_k) \\ f(x_k - \alpha d_k) &\leq f(x_k) + \nu_k + \theta_k - \rho\gamma_k\alpha^2 f(x_k)\end{aligned}$$

Proof. We show the statement by contraposition and define the function ϕ_+ as

$$\phi_+(t) = f(x_k + t d_k) - f(x_k) - \nu_k - \theta_k + \rho\gamma_k t^2 f(x_k)$$

Notice that $\phi_+(0) = -\nu_k - \theta_k < 0$ and $\phi_+(\alpha) > 0$ by hypothesis, since we assume that the line search is not satisfied by the step length α. From the Intermediate Value theorem, we know that there exists $t_+ \in (0, \alpha)$ such that

$$\phi_+(t_+) = f(x_k + t_+ d_k) - f(x_k) - \nu_k - \theta_k + \rho\gamma_k t_+^2 f(x_k) = 0 \tag{2.11}$$

Moreover, by *Lemma 2.3* ∇f is L-Lipschitz on $\text{conv}(\mathcal{L}_f(x_0))$ [1], thus we can bound the Taylor expansion of f as follow

$$\begin{aligned}f(x_k + t_+ d_k) &\leq f(x_k) + t_+ \nabla f(x_k)^T d_k + \frac{L t_+^2}{2}||d_k||^2 \\ f(x_k + t_+ d_k) - f(x_k) &\leq t_+ \nabla f(x_k)^T d_k + \frac{L t_+^2}{2}||d_k||^2 \end{aligned} \tag{2.12}$$

.

By combining (2.11) and (2.12), we obtain

$$\begin{aligned}t_+ \nabla f(x_k)^T d_k + \frac{L t_+^2}{2}||d_k||^2 - \nu_k - \theta_k + \rho\gamma_k t_+^2 f(x_k) \geq 0 \\ \frac{L t_+^2}{2}||d_k||^2 + \rho\gamma_k t_+^2 f(x_k) \geq -t_+ \nabla f(x_k)^T d_k\end{aligned}$$

[1] Note that we should prove formally that $x_k + t_+ d_k \in \text{conv}(\mathcal{L}_f(x_0))$ but it can be shown with equation (2.11) and a reasoning analogous to *Lemma 2.1*

Now, we introduce the definitions of the search direction $d_k = -\sigma_k E_k^T F(x_k)$ and the merit function $f(x) = \frac{1}{2}||F(x)||^2$ hence

$$\frac{Lt_+^2}{2}\,||\sigma_k E_k^T F(x_k)||^2 + \frac{\rho\gamma_k t_+^2}{2}||F(x_k)||^2 \geq \sigma_k t_+ \nabla f(x_k)^T E_k^T F(x_k)$$

Now, we recall that σ_k and γ_k are bounded, furthermore $E_k E_k^T = I$ hence

$$\begin{aligned}\frac{L\sigma_{\max}^2 + \rho\gamma_{\max}}{2}\,t_+||F(x_k)||^2 &\geq \sigma_k (E_k\nabla f(x_k))^T F(x_k)\\ t_+ &\geq \frac{2\sigma_k}{L\sigma_{\max}^2 + \rho\gamma_{\max}}\,\frac{(E_k\nabla f(x_k))^T F(x_k)}{||F(x_k)||^2}\end{aligned} \tag{2.13}$$

If we use a similar reasoning for

$$\phi_-(t) = f(x_k - td_k) - f(x_k) - \nu_k - \theta_k + \rho\gamma_k t^2 f(x_k)$$

We deduce that it exists $t_- \in (0, \alpha)$ with $\phi_-(t_-) = 0$ hence

$$t_- \geq -\frac{2\sigma_k}{L\sigma_{\max}^2 + \rho\gamma_{\max}}\,\frac{(E_k\nabla f(x_k))^T F(x_k)}{||F(x_k)||^2} \tag{2.14}$$

Finally, the claim follows from the fact that $\alpha > t_+$ and $\alpha > t_-$.
Therefore, by combining inequalities (2.13) and (2.14), we deduce that

$$\begin{aligned}\alpha &\geq \frac{2|\sigma_k|}{L\sigma_{\max}^2 + \rho\gamma_{\max}}\,\frac{|(E_k\nabla f(x_k))^T F(x_k)|}{||F(x_k)||^2}\\ &\geq \frac{2\sigma_{\min}}{L\sigma_{\max}^2 + \rho\gamma_{\max}}\,\frac{|(E_k\nabla f(x_k))^T F(x_k)|}{||F(x_k)||^2}\end{aligned}$$

□

We can observe that the main difference with Grapiglia and Chorobura [11] is that $(E_k\nabla f(x_k))^T F(x_k)$ appears instead of $\nabla f(x_k)^T F(x_k)$ meaning that we only consider a subset of the gradient's components. Thereafter, we might be interested in the quantity defined by

$$\boxed{\omega_k = \alpha_k \beta^{l_k} ||F(x_k)||} \tag{2.15}$$

Lemma 2.5. — *Let $(x_k)_{k=0}^{\infty}$ be a sequence generated by Algorithm 8, then we have*

$$\sum_{k=0}^{T-1} \omega_k^2 \leq \Omega \quad \textit{where} \quad \Omega = 2\,\frac{f(x_0) + \nu + \theta}{\rho\gamma_{\min}}$$

Proof. By the definition (2.15), the line search condition (2.7) can be written as

$$\begin{aligned}&f(x_{k+1}) \leq f(x_k) + \nu_k + \theta_k - \rho\gamma_k(\alpha_k\beta^{l_k})^2 f(x_k)\\ &\frac{\rho\gamma_k}{2}\,(\alpha_k\beta^{l_k})^2||F(x_k)|| \leq f(x_k) - f(x_{k+1}) + \nu_k + \theta_k\\ &\implies \quad \frac{\rho\gamma_{\min}}{2}\,\omega_k^2 \leq f(x_k) - f(x_{k+1}) + \nu_k + \theta_k\end{aligned}$$

Finally, taking the sum for $k = 0, 1, \ldots, T-1$ gives

$$\frac{\rho\gamma_{\min}}{2} \sum_{k=0}^{T-1} \omega_k^2 \leq f(x_0) - f(x_T) + \sum_{k=0}^{T-1} (\nu_k + \theta_k)$$
$$\implies \quad \sum_{k=0}^{T-1} \omega_k^2 \leq 2\, \frac{f(x_0) + \nu + \theta}{\rho\gamma_{\min}}$$

□

We are now ready to establish an upper bound on the required number of iterations to find an iterate x_k such that $\psi_k(x_k) \leq \epsilon$ with a given tolerance $\epsilon > 0$ and

$$\psi_k(x) = \begin{cases} \min\left(||F(x)||, \frac{|(E_k \nabla f(x))^T F(x)|}{||F(x)||}\right) & \text{if} \quad F(x) \neq 0 \\ 0 & \text{otherwise} \end{cases}$$

Notice that if $||F(x)|| \leq \epsilon$ then $\psi_k(x) \leq \epsilon$ but the converse is not true. Moreover, the function ψ_k which measures the accuracy of iterate x_k depends itself on k through the current matrix E_k.

Thereafter, we only consider two particular cases of *Algorithm 8* namely the Memoryless method and the Adaptive method. However, this can also be done for other choices as long as one can show that

$$\psi_k(x_k) > \epsilon \quad \implies \quad \omega_k \geq \tau\epsilon^p$$

for some constants $\tau \in \mathbb{R}$ and $p \in \mathbb{N}$. As a quick reminder, these methods differ only by the update formula for the guess α_{k+1}. The Memoryless method picks

$$\alpha_{k+1} = \alpha_0 \quad \text{for } k = 0, 1, 2, \ldots$$

while the Adaptive method takes

$$\alpha_{k+1} = \alpha_k \beta^{l_k - 1} \quad \text{for } k = 0, 1, 2, \ldots$$

2.5 Memoryless method

Thereafter, we prove in *Theorem 2.1* and *Corollary 2.1* that the Memoryless method has complexity bounds given by

$$\begin{aligned} T &= \mathcal{O}\left(\epsilon^{-2}\right) \\ V &= \mathcal{O}\left(\epsilon^{-2}\log(\epsilon^{-1})\right) \end{aligned} \tag{2.16}$$

where T, V are respectively the number of iterations and function evaluations to find x_T such that $\psi_T(x_T) \leq \epsilon$ with a tolerance $\epsilon \in (0, 1)$.

Lemma 2.6. — *Under the assumptions **A1**, **A2** and **A3**, let $(x_k)_{k=0}^{\infty}$ be a sequence generated by the Memoryless method. Suppose that for some $\epsilon \in (0,1)$ we have*

$$\psi_k(x_k) > \epsilon \quad for \quad k = 0, 1, \ldots, T-1$$

then for any $k \in \{0, 1, \ldots, T-1\}$

$$\omega_k = \alpha_k \beta^{l_k} ||F(x_k)|| \geq \tau_1 \epsilon \quad where \quad \tau_1 = \min(\alpha_0,\ \beta c)$$

Proof. The step length at iteration k is equal to $\alpha_0 \beta^{l_k}$, we consider two cases depending on the value of l_k. When $l_k = 0$, the claim is obviously true because

$$\omega_k = \alpha_0 ||F(x_k)|| > \alpha_0 \epsilon$$

Otherwise $l_k > 0$ and we can apply *Lemma 2.4* to $\alpha = \alpha_0 \beta^{l_k - 1}$ which can't be an admissible step size for the line search so

$$\begin{aligned} \alpha_0 \beta^{l_k - 1} > c\, \frac{|(E_k \nabla f(x_k))^T F(x_k)|}{||F(x_k)||^2} \geq c\, \frac{\psi_k(x_k)}{||F(x_k)||} \\ \implies \quad \alpha_0 \beta^{l_k} > \beta c \frac{\epsilon}{||F(x_k)||} \end{aligned}$$

Hence, we deduce that $\omega_k = \alpha_0 \beta^{l_k} ||F(x_k)|| \geq \beta c \epsilon$

□

Theorem 2.1. — *Under the assumptions **A1**, **A2** and **A3**, let $(x_k)_{k=0}^{\infty}$ be a sequence generated by the Memoryless method. Suppose that for some $\epsilon \in (0,1)$ we have*

$$\psi_k(x_k) > \epsilon \quad for \quad k = 0, 1, \ldots, T-1$$

then

$$T \leq \frac{\Omega}{\tau_1^2}\, \epsilon^{-2} = \mathcal{O}\left(\epsilon^{-2}\right)$$

Proof. From *Lemma 2.6*, we know that $\omega_k \geq \tau_1 \epsilon$ for $k = 0, 1, \ldots T-1$,
then we combine this with *Lemma 2.5*

$$\sum_{k=0}^{T-1} \omega_k^2 \leq \Omega \quad \implies \quad T \tau_1^2 \epsilon^2 \leq \Omega$$

Thus

$$T \leq \frac{\Omega}{\tau_1^2}\, \epsilon^{-2}$$

□

Corollary 2.1. — *Under the assumptions **A1**, **A2** and **A3**, let $(x_k)_{k=0}^{\infty}$ be a sequence generated by the Memoryless method. Suppose that for some $\epsilon \in (0,1)$, we find x_T the first iterate with $\psi_T(x_T) \leq \epsilon$ then the number of function evaluation V is bounded as follow*

$$V \leq 1 + \frac{\Omega}{\tau_1^2}\, \epsilon^{-2} \left(\frac{1}{\log \beta^{-1}} \log \left(\frac{\alpha_0 \sqrt{2(f(x_0) + \nu + \theta)}}{\tau_1 \epsilon} \right) + 1 \right) = \mathcal{O}\left(\epsilon^{-2} \log(\epsilon^{-1})\right)$$

Proof. By assumption $\psi_k(x_k) > \epsilon$ for $k = 0, 1, \ldots, T-1$ so by *Lemma 2.6*

$$\alpha_0 \beta^{l_k} ||F(x_k)|| \geq \tau_1 \epsilon \quad \Longrightarrow \quad \beta^{l_k} \geq \frac{\tau_1 \epsilon}{\alpha_0 ||F(x_k)||} \tag{2.17}$$

Moreover, as a consequence of *Lemma 2.1*

$$||F(x_k)|| = \sqrt{2f(x_k)} \leq \sqrt{2(f(x_0) + \nu + \theta)} \tag{2.18}$$

Then, we use the results (2.18) in inequality (2.17) and take logarithm on both side

$$\begin{aligned} l_k \log \beta &\geq \log \left(\frac{\tau_1 \epsilon}{\alpha_0 \sqrt{2(f(x_0) + \nu + \theta)}} \right) \\ l_k &\leq \frac{1}{\log \beta^{-1}} \log \left(\frac{\alpha_0 \sqrt{2(f(x_0) + \nu + \theta)}}{\tau_1 \epsilon} \right) \end{aligned} \tag{2.19}$$

At the end, we combine (2.19) with *Theorem 2.1* in order to bound the number of function evaluations

$$\begin{aligned} V &\leq 1 + 2 \sum_{k=0}^{T-1} (l_k + 1) \\ &\leq 1 + 2T \left(\frac{1}{\log \beta^{-1}} \log \left(\frac{\alpha_0 \sqrt{2(f(x_0) + \nu + \theta)}}{\tau_1 \epsilon} \right) + 1 \right) \end{aligned}$$

□

2.6 Adaptive method

Thereafter, we prove in *Theorem 2.2* and *Corollary 2.2* that the Adaptive method has complexity bounds given by

$$\boxed{\begin{aligned} T &= \mathcal{O}\left(\epsilon^{-4}\right) \\ V &= \mathcal{O}\left(\epsilon^{-4}\right) \end{aligned}} \tag{2.20}$$

where T, V are respectively the number of iterations and function evaluations to find x_T such that $\psi_T(x_T) \leq \epsilon$ with a tolerance $\epsilon \in (0, 1)$. At the first look, the theoretical bounds (2.20) of the Adaptive method is worst compared to the bounds (2.16) of the Memoryless method. We can't get the same result as presviously because of a small detail in the proof namely the case $l_{k+1} = 0$ in *Lemma 2.7*. However, it shows that the number of function evaluations is bounded and the method may behave well in practice.

Lemma 2.7. — *Under the assumptions **A1**, **A2** and **A3**, let $(x_k)_{k=0}^{\infty}$ be a sequence generated by the Adaptive method. Suppose that for some $\epsilon \in (0, 1)$ we have*

$$\psi_k(x_k) > \epsilon \quad \textit{for} \quad k = 0, 1, \ldots, T-1$$

then for any $k \in \{0, 1, \dots, T-1\}$

$$\omega_k = \alpha_k \beta^{l_k} ||F(x_k)|| \geq \tau_2 \epsilon^2 \quad \textit{where} \quad \tau_2 = \min\left(\alpha_0, \frac{\beta c}{\sqrt{2(f(x_0) + \nu + \theta)}}\right)$$

Proof. Since we assume that $\psi_k(x_k) > \epsilon$, we have also $||F(x_k)|| > \epsilon$. Therefore, it is sufficient to prove

$$\alpha_k \beta^{l_k} \geq \tau_2 \epsilon \quad \text{for} \quad k = 0, 1, \dots, T-1 \tag{2.21}$$

We proceed by induction, the case $k = 0$ given by $\alpha_0 \beta^{l_0} \geq \tau_2 \epsilon$ is obviously true for $l_0 = 0$ otherwise we use a reasoning like (2.22). Then, we assume that the inequality (2.21) holds. We consider two cases depending on the value of l_{k+1}. When $l_{k+1} = 0$, we find by the induction hypothesis

$$\alpha_{k+1} \beta^{l_{k+1}} = \alpha_{k+1} = \alpha_k \beta^{l_k - 1} \geq \alpha_k \beta^{l_k} \geq \tau_2 \epsilon$$

Otherwise $l_{k+1} > 0$ and we can apply *Lemma 2.4* to $\alpha = \alpha_{k+1} \beta^{l_{k+1}-1}$ which can't be an admissible step size for the line search so

$$\begin{aligned} \alpha_{k+1} \beta^{l_{k+1}-1} > c \frac{|(E_k \nabla f(x_k))^T F(x_k)|}{||F(x_k)||^2} \geq c \frac{\psi_k(x_k)}{||F(x_k)||} \\ \implies \quad \alpha_{k+1} \beta^{l_{k+1}} > \beta c \frac{\epsilon}{||F(x_k)||} \end{aligned}$$

With a similar trick as (2.18), we deduce that

$$\alpha_{k+1} \beta^{l_{k+1}} > \frac{\beta c}{\sqrt{2(f(x_0) + \nu + \theta)}} \epsilon \geq \tau_2 \epsilon \tag{2.22}$$

□

Theorem 2.2. — *Under the assumptions* ***A1***, ***A2*** *and* ***A3***, *let* $(x_k)_{k=0}^{\infty}$ *be a sequence generated by the Adaptive method. Suppose that for some* $\epsilon \in (0, 1)$ *we have*

$$\psi_k(x_k) > \epsilon \quad \textit{for} \quad k = 0, 1, \dots, T-1$$

then

$$T \leq \frac{\Omega}{\tau_2^2} \epsilon^{-4} = \mathcal{O}\left(\epsilon^{-4}\right)$$

Proof. From *Lemma 2.7*, we know that $\omega_k \geq \tau_2 \epsilon^2$ for $k = 0, 1, \dots T-1$, then we combine this with *Lemma 2.5*

$$\sum_{k=0}^{T-1} \omega_k^2 \leq \Omega \quad \implies \quad T \tau_2^2 \epsilon^4 \leq \Omega$$

Thus

$$T \leq \frac{\Omega}{\tau_2^2} \epsilon^{-4}$$

□

Corollary 2.2. — *Under the assumptions* ***A1***, ***A2*** *and* ***A3***, *let* $(x_k)_{k=0}^{\infty}$ *be a sequence generated by the Adaptive method. Suppose that for some* $\epsilon \in (0, 1)$, *we find* x_T *the first iterate*

with $\psi_T(x_T) \leq \epsilon$ then the number of function evaluation V is bounded as follow

$$N \leq 1 + 4\frac{\Omega}{\tau_2^2}\,\epsilon^{-4} + \frac{2}{\log \beta^{-1}} \log\left(\frac{\alpha_0 \beta}{\tau_2}\epsilon^{-1}\right) = \mathcal{O}\left(\epsilon^{-4}\right)$$

Proof. Applying logarithm to both sides of the update formula gives

$$\alpha_{k+1} = \alpha_k \beta^{l_k - 1} \qquad \Longrightarrow \qquad (l_k - 1)\log\beta = \log\alpha_{k+1} - \log\alpha_k$$

If we take the sum for $k = 0, 1, \ldots, T-1$ then we find

$$\log\beta \sum_{k=0}^{T-1} (l_k - 1) = \log\alpha_T - \log\alpha_0$$

Since we assume that $\psi_k(x_k) > \epsilon$ for $k = 0, 1, \ldots, T-1$,
we can write thanks to the inequality (2.21)

$$\alpha_T = \alpha_{T-1}\beta^{l_{T-1}-1} \geq \frac{\tau_2 \epsilon}{\beta}$$

Therefore

$$\begin{aligned} \log\beta \sum_{k=0}^{T-1} (l_k - 1) &\geq \log\left(\frac{\tau_2\epsilon}{\alpha_0\beta}\right) \\ \sum_{k=0}^{T-1} l_k &\leq T + \frac{1}{\log\beta^{-1}} \log\left(\frac{\alpha_0\beta}{\tau_2\epsilon}\right) \end{aligned} \tag{2.23}$$

At the end, we combine (2.23) with *Theorem 2.2* in order to bound the number of function evaluations

$$\begin{aligned} V &\leq 1 + 2\sum_{k=0}^{T-1} (l_k + 1) \\ &= 1 + 2T + 2\sum_{k=0}^{T-1} l_k \\ &\leq 1 + 4T + \frac{2}{\log\beta^{-1}} \log\left(\frac{\alpha_0\beta}{\tau_2\epsilon}\right) \end{aligned}$$

□

Chapter 3

Numerical experiments

Now that we have obtained theoretical bounds, we look at three methods belonging to the class described in Chapter 2 and observe how they actually behave numerically on test problems.

3.1 Implementation

In all the tests, the methods are implemented in MATLAB (version R2022a) and the experiments are made with a processor Intel Core i5.

We use the following set of parameters

$$\begin{array}{cccc} \alpha_0 = 1 & \beta = 0.5 & \rho = 10^{-4} & \epsilon = 10^{-6} \\ \sigma_0 = 1 & \sigma_{\min} = 10^{-1} & \sigma_{\max} = 10^{10} & \gamma_k = 1 \end{array}$$

Furthermore, we might sometimes give the number of function evaluations obtained when we pick $\gamma_k = 2\sigma_k^2$ as in Echebest, Schuverdt, and Vignau [9]. Nevertheless we will see that the choice $\gamma_k = 1$ makes that the methods behave generally better on the considered test set. Both resulting line search are reminded below

$$\begin{aligned} \gamma_k = 1 \quad &\Longrightarrow \quad f(x_{k+1}) \leq f(x_k) + \nu_k + \theta_k - \rho\alpha^2 f(x_k) \\ \gamma_k = 2\sigma_k^2 \quad &\Longrightarrow \quad f(x_{k+1}) \leq f(x_k) + \nu_k + \theta_k - \rho\alpha^2 ||d_k||^2 \end{aligned}$$

As described in Chapter 2, the spectral coefficient is computed as follow

$$\sigma_{k+1} = \frac{(E_{k+1}s_k)^T(E_{k+1}s_k)}{(E_{k+1}s_k)^T y_k} \tag{3.1}$$

where $s_k = x_{k+1} - x_k$ and $y_k = F(x_{k+1}) - F(x_k)$. If the value provided by formula (3.1) doesn't satisfy $\sigma_{\min} \leq |\sigma_k| \leq \sigma_{\max}$ then we use the same rule as Cruz, Martínez, and Raydan [6] and take instead

$$\sigma_k = \begin{cases} 1 & \text{if} \quad ||F(x_k)|| > 1 \\ 10^5 & \text{if} \quad ||F(x_k)|| < 10^{-5} \\ ||F(x_k)||^{-1} & \text{otherwise} \end{cases}$$

For the moment, we choose to keep free the definition of the sequence $(E_k)_{k=0}^{\infty}$ in order to observe later which options seem the best based on the numerical experiments. Once all these parameters are fixed, a particular method belonging to the class generated by *Algorithm 8* is fully characterized by the sequences $(\alpha_k)_{k=0}^{\infty}$, $(\nu_k)_{k=0}^{\infty}$ and $(\theta_k)_{k=0}^{\infty}$.

We focus our analysis on the following algorithms

✧ **DFSAUNE** (abbreviated as DFS)

$$\alpha_k = \alpha_0 \qquad \nu_k = -f(x_k) + \max_{\min(0,k-M+1)\leq j\leq k} f(x_j) \qquad \theta_k = \frac{\epsilon}{2^k} \qquad M = 2$$

✧ **Memoryless method** (abbreviated as MLS)

$$\alpha_k = \alpha_0 \qquad \nu_k = 0 \qquad \theta_k = \frac{\epsilon}{2^k}$$

✧ **Adaptive method** (abbreviated as ADP)

$$\alpha_{k+1} = \alpha_k \beta^{l_k - 1} \qquad \nu_k = 0 \qquad \theta_k = \frac{\epsilon}{2^k}$$

3.2 Methodology

On one hand, the results are presented in tables containing the number of function evaluations and the best value of $||F(x_k)||$ achieved by the algorithm (given this number of function evaluations). The size of the problem (m, n) is also provided and one can find the references to the exact formulation in *Appendix A*.

In the tables, we consider that a problem is solved at iteration k when we have

$$||F(x_k)|| \leq \max\left(1,\ ||F(x_0)||\right)\ \epsilon \tag{3.2}$$

The purpose of this stopping criterion is that if $||F(x_0)||$ is very large then we will be satisfied by reducing only the relative error $\frac{||F(x_k)||}{||F(x_0)||}$. Otherwise we will look at the absolute error. We also require that the total number of function evaluations done by the methods must be less than

$$\texttt{MAXEval} = 30000$$

If a problem is not solved in the sense of (3.2) within this budget of function evaluations then we write `MAXEval` in the corresponding column.

On the other hand, we would like to compare the algorithms relatively to each other with data profiles, a tool introduced by Moré and Wild [17]. We consider a set of solver $\mathcal{S}$ that we want to test on some set of problems $\mathcal{P}$. Given a point x^s generated by the solver $s \in \mathcal{S}$, a problem $p \in \mathcal{P}$ defined by $F^p(x) = 0$ with $f^p(x) = \frac{1}{2}\ ||F^p(x)||^2$ and a maximal budget $t_{\max}$, we define the stopping criterion

$$f^p(x_0) - f^p(x^s) \geq (1-\eta)\left(f^p(x_0) - f^p_{\text{best}}\right) \tag{3.3}$$

where $\eta \in (0,1)$ and f^p_{best} correspond to the best value of $f(x)$ obtained by any solver in $\mathcal{S}$ for problem p with a budget of $t_{\max}$ function evaluations. The inequality (3.3) can also be written under the form

$$\begin{aligned} & f^p(x^s) \leq f^p_{\text{best}} + \eta\left(f^p(x_0) - f^p_{\text{best}}\right) \\ \Longrightarrow \quad & \frac{f^p(x^s) - f^p_{\text{best}}}{f^p(x_0) - f^p_{\text{best}}} \leq \eta \end{aligned}$$

Thus, the stopping criterion (3.3) can be interpreted as requiring the relative decrease to be smaller than some tolerance η when we take as solution the best point provided by all the solvers in $\mathcal{S}$. Notice that this criterion depends highly on the set of solvers considered. For instance, if all the solvers behave badly on a problem $p \in \mathcal{P}$ then f^p_{best} can be very far from $f(x^\star) = 0$ but some solvers might still reach the stopping criterion. That is why we first need to observe the behavior of the methods with the stopping criterion (3.2). Then, we compare them according to (3.3).

If we note $t_{p,s}$ the required number of function evaluations for solver s to find a point x^s such that (3.3) holds for problem p then the data profile curve of s is given by

$$d_s(t) = \frac{1}{|\mathcal{P}|} \left|\{p \in \mathcal{P} \mid t_{p,s} \leq t\}\right|$$

In other words, $d_s(t)$ is the proportion of problems such that the solver $s \in \mathcal{S}$ reaches the stopping criterion in less than t function evaluations. All the data profile graphs are done with the values $\eta = 10^{-5}$ and $t_{\max} = 1000$.

3.3 Results of the tests

❒ **First test set**

Table 3.1 summarizes the results obtained for the same test set as Grapiglia and Chorobura [11] which contains 16 problems with a number of unknowns ranging from 2 to 40. The problems were designed by Moré, Garbow, and Hillstrom [16] to put more emphasis on the reliability and the robustness of algorithms. Notice that all these tests are such that $n = m$ so we have $E_k = I$ for all k.

Problem	n=m	# Evaluations			$\min_k \|\|F(x_k)\|\|$		
		DFS	MLS	ADP	DFS	MLS	ADP
1	2	MAXEval	MAXEval	MAXEval	1.8735e+00	1.8112e+00	**1.5791e+00**
2	2	MAXEval	**3663**	MAXEval	1.1782e+01	1.1635e-05	1.1297e+01
3	2	MAXEval	MAXEval	MAXEval	**1.1196e-01**	1.1446e-01	3.4622e-01
4	3	MAXEval	MAXEval	MAXEval	**4.2500e+01**	**4.2500e+01**	**4.2500e+01**
5	4	MAXEval	MAXEval	MAXEval	7.0768e+00	6.8793e+00	**6.4302e+00**
6	10	MAXEval	MAXEval	MAXEval	4.1894e+00	4.0500e+00	**3.5309e+00**
7	12	MAXEval	MAXEval	MAXEval	1.2257e+01	1.1915e+01	**1.1137e+01**
8	10	**11**	12	60 (57)	1.0045e-07	9.9205e-08	3.7435e-08
9	40	17707	MAXEval	**3647**	1.2773e-04	1.0590e+01	1.2735e-04
10	10	706	920	**345**	9.2917e-07	9.5521e-07	8.9314e-07
11	10	**6**	**6**	42 (89)	1.2296e-07	1.2296e-07	8.0665e-07
12	10	**59**	MAXEval	MAXEval	4.2521e-06	9.0420e-01	9.8942e-01
13	10	MAXEval	MAXEval	MAXEval	2.7949e+00	**1.8231e+00**	1.9108e+00
14	10	**3**	**3**	**3**	0.0000e+00	0.0000e+00	0.0000e+00
15	10	**19**	**19**	26	1.4945e-13	1.4945e-13	4.2307e-14
16	10	**17**	**17**	23	3.4436e-14	3.4436e-14	7.2492e-14

TABLE 3.1: Results obtained with the same test set as Grapiglia and Chorobura [11]

If the number of function evaluations obtained with $\gamma_k = 2\sigma_k^2$ is different, we add it to the column "#Evaluations" between parenthesis but $\min_k ||F(x_k)||$ is always computed only for $\gamma_k = 1$. The best results with respect to the number of function evaluations are **highlighted** . In the case where the problem is not solved, we look for the best method in terms of $\min_{0 \leq k \leq \texttt{MAXEval}} ||F(x_k)||$.

We observe in *Table 3.1* that the problems $1 \rightarrow 7$ and $12 \rightarrow 13$ are very hard to solve with *Algorithm 8* because at least two of the presented methods don't succeed to solve them in the given budget $\texttt{MAXEval} = 30000$. Nevertheless, the problems 2 and 12 are respectively solved by the Memoryless method and DF-SAUNE with a number of function evaluations relatively small compared to $\texttt{MAXEval}$.

Notice that picking $\gamma_k = 1$ or $\gamma_k = 2\sigma_k^2$ has almost no effect on the result. This is due to the fact that $\rho = 10^{-4}$ is very small hence $-\rho\alpha^2 f(x_k)$ and $-\rho\alpha^2||d_k||^2$ are small too and the line search procedure picks the same step size for both in general. There are two exceptions occurring for the Adaptive method namely the problems 8 and 11. In the first case $\gamma_k = 2\sigma_k^2$ saves 3 function evaluations but the second case doubles it hence we are more prone to choose $\gamma_k = 1$.

Overall, DF-SAUNE reaches with less function evaluations the stopping criterion closely followed by the Memoryless method. Nevertheless, they are widely beaten by the Adaptive method in problems 9 and 10. Moreover, for the non solved problems, the values of $||F(x_k)||$ found by the Adaptative method are usually slightly better. This behavior is a consequence of the way we guess the first step size α_{k+1} in the line search. Indeed, the Adaptive method does most of the time less function evaluations by iteration. It means more iterations which is an advantage when the step sizes are not too small compared to the other methods.

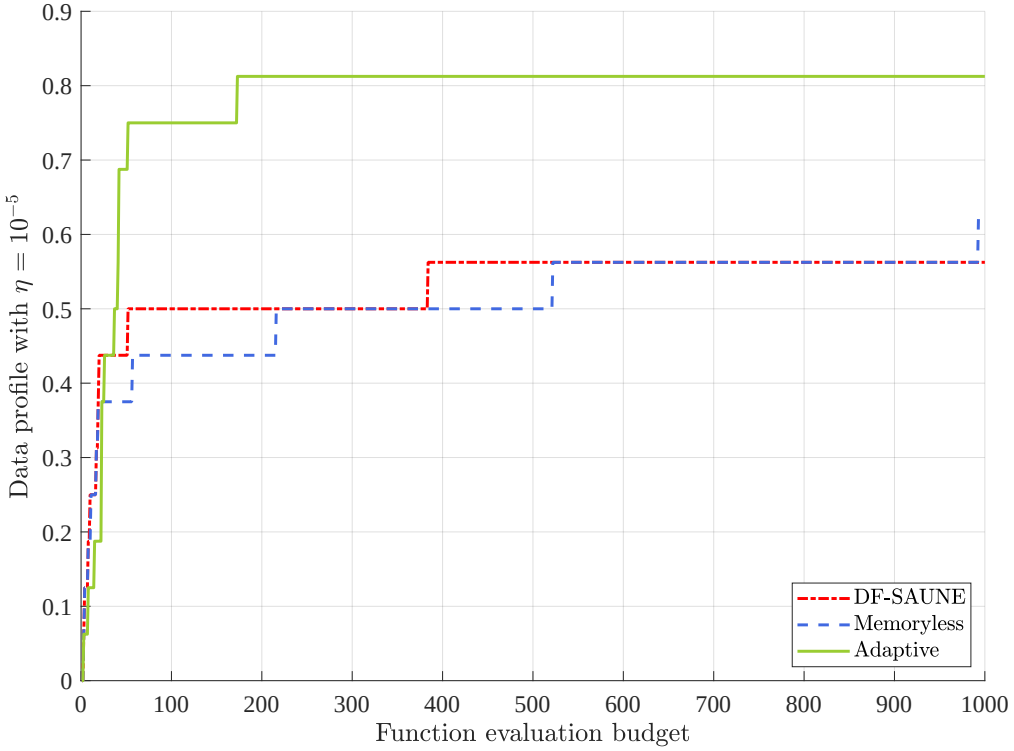


FIGURE 3.1: Data profile with the same test set as Grapiglia and Chorobura [11]

Figure 3.1 represents the data profiles corresponding to the problems from *Table 3.1*. Contrary to previously, we focus here on performances comparison between the methods with a lower budget, 1000 instead of 30000 function evaluations. It is worth to mention that we look at the decrease of the objective regardless if the problem was solved. One can notice that the Adaptive method reaches after approximately 200 function evaluations, a proportion of 80% problems solved for the relative stopping criterion (3.3). The data profiles of the Memoryless method and DF-SAUNE are almost identical and struggle to solve more than 60% of the problems with this shorter budget.

However, we remark in *Figure 3.1* that about half of the problems are considered as solved with about 50 function evaluations by the three methods. It means that some problems are solved quite quickly according to the criterion (3.3) and *Algorithm 8* encounters some difficulties for the remaining. Finally, we see that for very few function evaluations (less than 30), it is better to use DF-SAUNE or the Memoryless method.

❒ Second test set

Thereafter, we present in *Table 3.2* the result for the tests found in Echebest, Schuverdt, and Vignau [9] which contains 20 small size problems and 6 larger problems. Most of these problems are underdetermined hence we first try the standard strategy for defining the sequence $(E_k)_{k=0}^{\infty}$.

Problem	m, n	# Evaluations			$\min_k \|\|F(x_k)\|\|$		
		DFS	MLS	ADP	DFS	MLS	ADP
17	1, 2	**4**	**4**	**4**	4.4409e-15	4.4409e-15	4.4409e-15
18	1, 2	73	**23**	55	1.0445e-05	1.6109e-05	2.2841e-05
19	2, 2	**23**	31	55	7.6680e-06	1.7169e-05	1.6903e-05
20	1, 3	104	**86**	100	1.8861e-05	9.4628e-06	9.4628e-06
21	1, 3	**2**	**2**	**2**	0.0000e+00	0.0000e+00	0.0000e+00
22	2, 4	**114**	175	759	1.5376e-06	9.5516e-06	9.9879e-06
23	3, 4	**248**	385	288	5.5557e-07	7.3671e-07	9.9024e-07
24	2, 4	**2**	**2**	**2**	0.0000e+00	0.0000e+00	0.0000e+00
25	2, 5	538	**104**	187	2.2005e-07	8.6495e-07	4.7379e-07
26	3, 5	194	157	**92**	6.5183e-07	3.1195e-07	2.7550e-07
27	2, 5	**10**	11	13	0.0000e+00	5.0243e-15	5.0243e-15
28	3, 5	**2**	**2**	**2**	0.0000e+00	0.0000e+00	0.0000e+00
29	4, 7	**137**	168	237	1.8489e-06	2.9735e-06	4.5840e-06
30	2, 3	435	476	**132**	6.3515e-06	1.2485e-05	1.2815e-05
31	2, 3	216	**110**	135	1.2693e-05	7.0179e-06	1.3071e-05
32	2, 5	507	**117**	14778	1.9936e-05	5.0651e-05	5.6559e-05
33	3, 5	**1205**	3823	`MAXEval`	4.5273e-06	4.0327e-06	1.1162e+00
34	3, 5	**428**	2218	`MAXEval`	6.0454e-06	8.0510e-06	1.5187e+00
35	3, 5	**571**	`MAXEval`	`MAXEval`	4.0657e-06	3.0523e+00	3.9796e+00
36	3, 10	**502**	578 (711)	2071	1.0769e-06	1.2926e-06	1.1740e-06
37	300, 300	295 (296)	199 (`MAXEval`)	**150**	5.0744e-03	1.6436e-03	1.9924e-03
38	300, 300	111	**91**	344	3.6946e+01	2.7445e+01	3.6307e+01
39	150, 300	**4**	**4**	**4**	2.2204e-16	2.2204e-16	2.2204e-16
40	150, 300	244	292	**96**	2.7240e-01	2.2125e-01	1.5093e-01
41	149, 150	207 (209)	**146** (`MAXEval`)	167 (158)	2.6999e-06	2.4541e-05	1.2572e-05
42	299, 300	191 (216)	**80**	`MAXEval`	1.3997e-05	2.3543e-05	3.7580e-01

TABLE 3.2: Results obtained with the same test set as Echebest, Schuverdt, and Vignau [9]

We see that these tests are a bit easier than the previous one as all the problems are at least solved by one of the method. Although it still gives a good idea of the behavior of the methods, we need to be more critical. Firstly, notice that the choice of γ_k seems to be more important for large size problems but $\gamma_k = 1$ remains the best in most of the tests. Especially, the Memoryless method with $\gamma_k = 2\sigma_k^2$ can't solve the problems 37 and 41 while it can solve them with $\gamma_k = 1$. As said in Echebest, Schuverdt, and Vignau [9], the methods for underdetermined system of equations using this sequence $(E_k)_{k=0}^{\infty}$ are in general more efficient when n can be divided by m because the partitioning of the variables is done equally. That is to say that the last matrix in each period E_{N-1} doesn't introduce a bias by reusing some of the first variables.

Few test problems (17, 21, 24, 28 and 39) that have a linear or quadratic structure are solved quite easily by the three methods. However, it is not always the case for *Algorithm 8*. For instance, problems 39 and 40 have the same function $F(x)$ but the initial point in the problem 40 is chosen very far from the solution hence a significant increase in the number of function evaluations. From a global view point, the best methods in terms of function evaluations are DF-SAUNE in 9 problems, Memoryless method in 8 problems and Adaptative method in 4 problems (with equality for the other problems).

One can note that the Adaptive method works less efficiently in the underdetermined case. The main reason is that at each iteration, the set of variables updated by the step changes. Therefore, using the information from the previous step is not always convenient as it might be very small compared to other admissible step sizes.

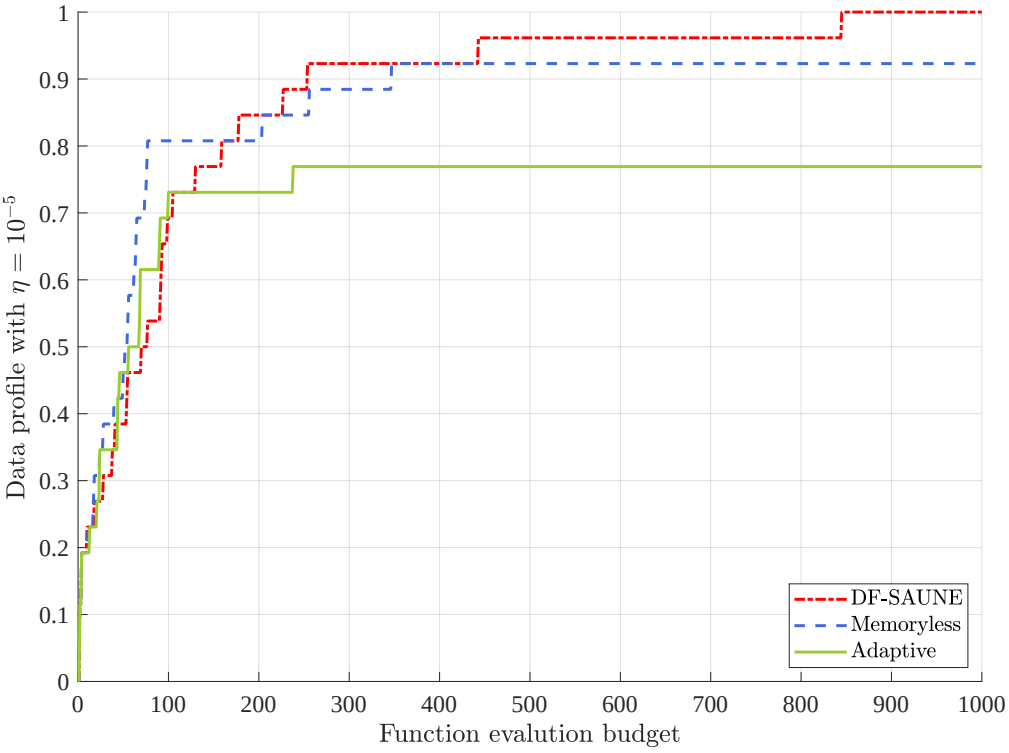


FIGURE 3.2: Data profile for the test set from Echebest, Schuverdt, and Vignau [9]

We notice in *Figure 3.2* that DF-SAUNE becomes the best method on this set of problems after 150 function evaluations as always closely followed by the Memoryless method. Even if the performance of the Adaptive method gets worse with underdetermined systems of equations, we are barely at 80% of the problems solved which is still satisfactory.

❒ **Third test set**

Now, we compare the result with the standard way of building $(E_k)_{k=0}^{\infty}$ and the modulo approach, see *Chapter 2*. We denote methods using the standard sequence by "(0)" and the modulo sequence by "(%)". Since both these sequences are identical when modulo$(m, n) \neq 0$, we only consider a subset of problems from Echebest, Schuverdt, and Vignau [9] and we again **highlight** the most efficient strategy for the number of function evaluations among the two.

Problem	m, n	# Evaluations					
		DFS(0)	DFS(%)	MLS(0)	MLS(%)	ADP(0)	ADP(%)
23	3, 4	**248**	285	385	**254**	288	**162**
25	2, 5	538	**104**	**104**	174	187	**155**
26	3, 5	194	**167**	**157**	186	**92**	261
27	2, 5	**10**	256	**11**	100	**13**	121
28	3, 5	**2**	**2**	**2**	**2**	**2**	**2**
29	4, 7	**137**	1582	**168**	1936	**237**	MAXEval
30	2, 3	435	**213**	476	**187**	**132**	156
31	2, 3	216	**107**	**110**	226	135	**99**
32	2, 5	507	**434**	**117**	486	**14778**	MAXEval
33	3, 5	1205	**476**	3823	**417**	MAXEval	**464**
34	3, 5	**428**	536	2218	**287**	MAXEval	1354
35	3, 5	**571**	739	MAXEval	**427**	MAXEval	**3136**
36	3, 10	**502**	828	578	**289**	**2071**	MAXEval
41	149, 150	**207**	MAXEval	**146**	MAXEval	**167**	MAXEval
42	299, 300	**191**	MAXEval	**80**	MAXEval	MAXEval	MAXEval

TABLE 3.3: Subset of tests with modulo$(m, n) \neq 0$ from Echebest, Schuverdt, and Vignau [9]

We can't really identify from the *Table 3.3*, a strategy which is clearly more efficient than the other. Despite this, we observe that some problems are solved only for one type of sequence $(E_k)_{k=0}^{\infty}$. For instance, the problems 41 and 42 are unsolved for the modulo strategy (with any methods) but they are mostly solved with the standard strategy. We have the contrary for the Memoryless method applied to problem 35 and the Adaptive method applied to problems 33, 34 and 35. The main issue with the modulo strategy is that the periodic sequence $E_0, E_1, \ldots, E_{N-1}$ might contain many matrices. This is typically the case for the sequences generated for the problems 41, 42 and 29 which can have a very large value for N while the standard way has always $N = \lceil \frac{n}{m} \rceil$.

The corresponding data profiles are shown below in *Figure 3.3*, *Figure 3.4*, *Figure 3.5* for respectively DF-SAUNE, the Memoryless method and the Adaptive method. Concerning DF-SAUNE and the Memoryless method, we remark that the standard strategy works better on the full test. Nevertheless, we think that the data profiles are a bit biased by the problems 41 and 42 which are quite hard to solve with the modulo strategy. Therefore, the modulo strategy should lead to an improvement for the Adaptative and the Memoryless methods when the period N of the sequence is not too important.

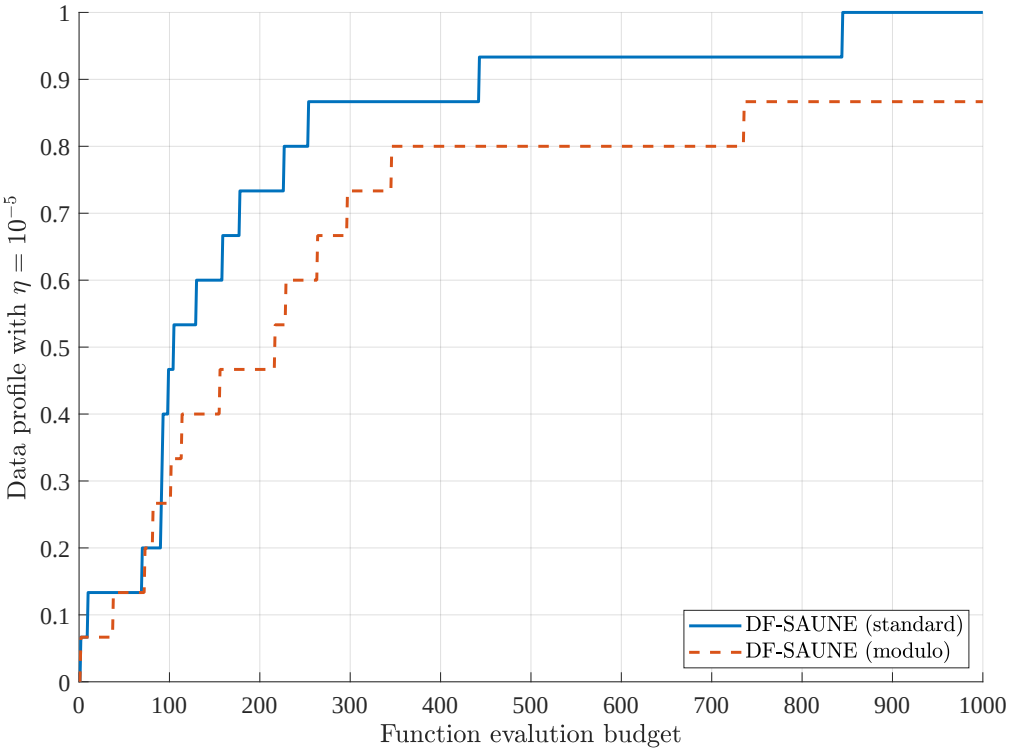


FIGURE 3.3: Data profile for a subset of tests with modulo$(m, n) \neq 0$ from Echebest, Schuverdt, and Vignau [9]

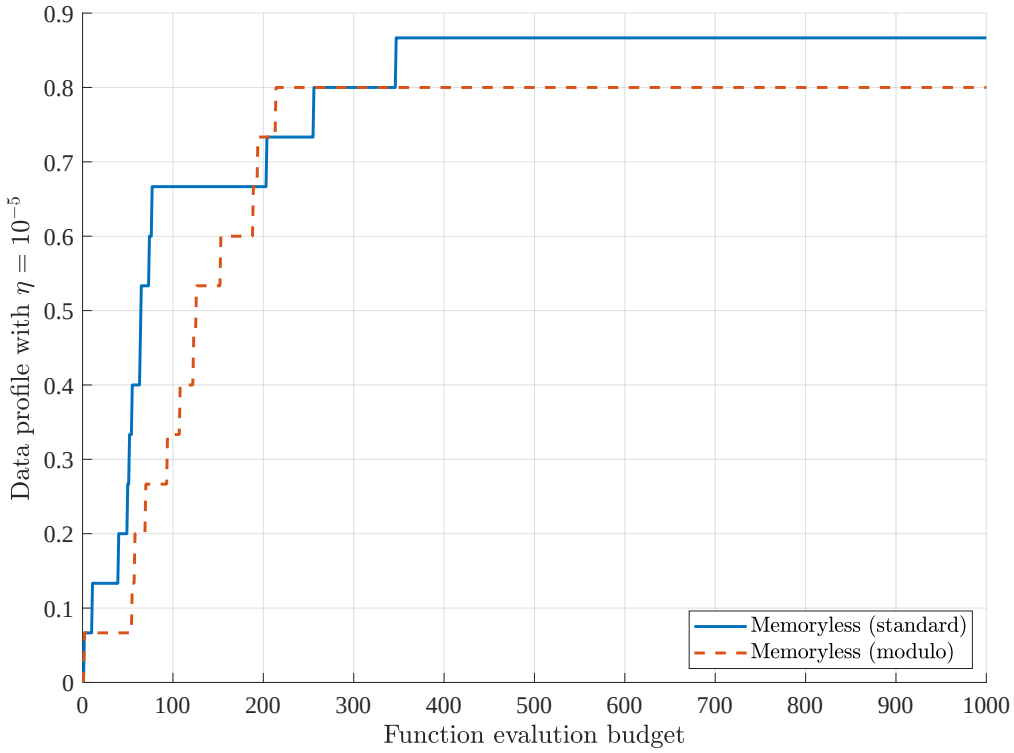


FIGURE 3.4: Data profile for a subset of tests with modulo$(m, n) \neq 0$ from Echebest, Schuverdt, and Vignau [9]

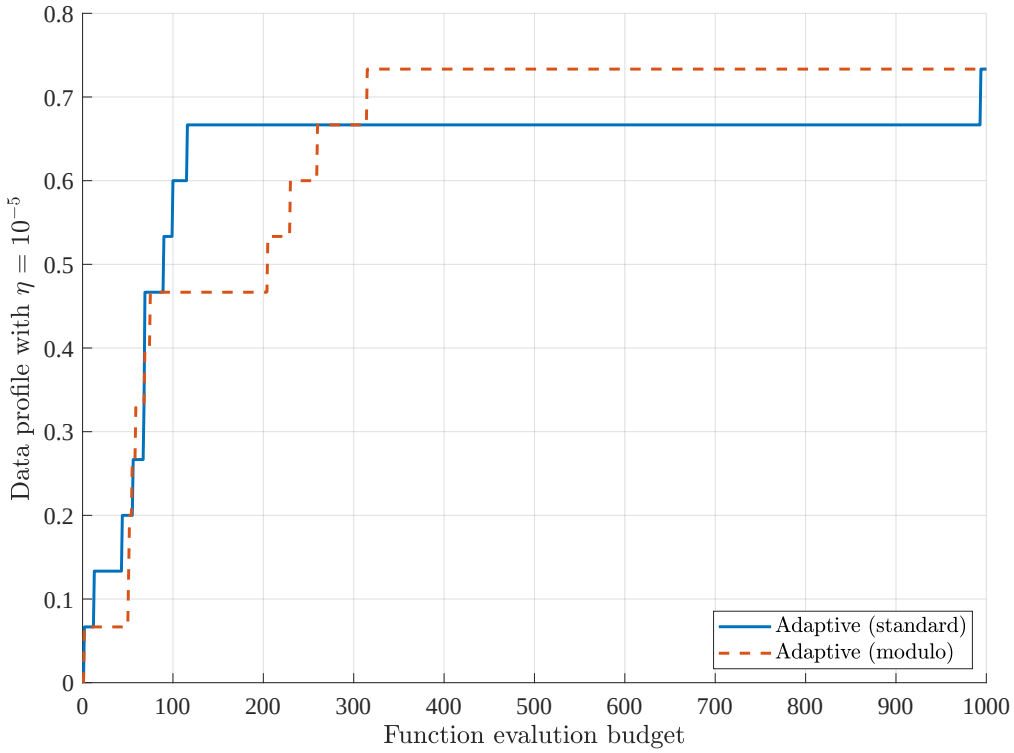


FIGURE 3.5: Data profile for a subset of tests with modulo$(m, n) \neq 0$ from Echebest, Schuverdt, and Vignau [9]

Chapter 4

Adversarial attack

This chapter presents an application of derivative-free methods for solving underdetermined systems of equations in the context of adversarial attack against deep neural networks.

4.1 Context

Nowadays, the deep neural networks find a growing number of applications such as self-driving cars, facial recognition, email malware detection or forecasting of natural disasters to cite only few of them. Nevertheless, it was shown that in some cases these classifiers suffer from a lack of robustness. Namely, tiny modifications of the input might lead to large variations of the output.

It is particularly problematic when a self-driving car is not able to interpret the perturbed image of a road sign or an email containing a malware is written such that it is classified as safe. Especially, in the context of image classification slight changes even not visible for the human being, could produce the misclassification of initially well-classified images.

Formally, we consider an input space $\mathcal{X}$, a discrete space $\mathcal{C}$ containing all the classes and a function $c : \mathcal{X} \to \mathcal{C}$ assigning to each $x \in \mathcal{X}$ its true class $c(x)$.
For example :

- ✧ Image classification
  $\mathcal{X} = \mathbb{R}^{M \times N \times 3}$ images with $M \times N$ pixels
  $\mathcal{C} = \{"Panda", "Apple", "Llama", "Tree", \dots\}$
  $c(x)$ gives what really represents the image
- ✧ Email malware detection
  $\mathcal{X} =$ subsets of words in $\{"Hello", "Dear", "Sincerely", \dots\}$
  $\mathcal{C} = \{"Dangerous", "Safe"\}$
  $c(x)$ says if the email contains truly a malware

In this setup, a classifier is a function $\hat{c} : \mathcal{X} \to \mathcal{C}$ assigning to each $x \in \mathcal{X}$ a guess in $\mathcal{C}$ such that $\hat{c}(x) \approx c(x)$. We say that $x \in \mathcal{X}$ is misclassified when $\hat{c}(x) \neq c(x)$. Obviously, a "good" classifier should be such that most of the elements in $\mathcal{X}$ are well-classified. As we mainly focus on neural network classifiers, we briefly remind how the predictor $\hat{c}(x)$ is obtained in this case.

A neural network is usually represented by a weighted graph which can be divided in successive stages called layer. At the begining, we have the input layer $x \in \mathbb{R}^n$ then the information is propagated through the hidden layers. Between each layer, we apply some kind of algebraic manipulations from one node to another node of

the next layer, see *Figure 4.1*. When we want to emphasis on the fact that there are many hidden layers, we often speak about deep neural network. Finally, the last layer is the output $y(x) \in \mathbb{R}^m$ which is usually a conditional class probability in the case of neural networks trained for classification. It means that we have for $i = 1, \dots, m$

$$y_i(x) = P(c_i|x) = \text{"probability of observing the class } c_i \text{ given the input } x\text{"}$$

Afterwards, the classifier is obtained by looking at the output of the neural network and picking the class with the biggest conditional probability hence

$$\hat{c}(x) = \arg\max_{c \in \mathcal{C}} P(c|x) \tag{4.1}$$

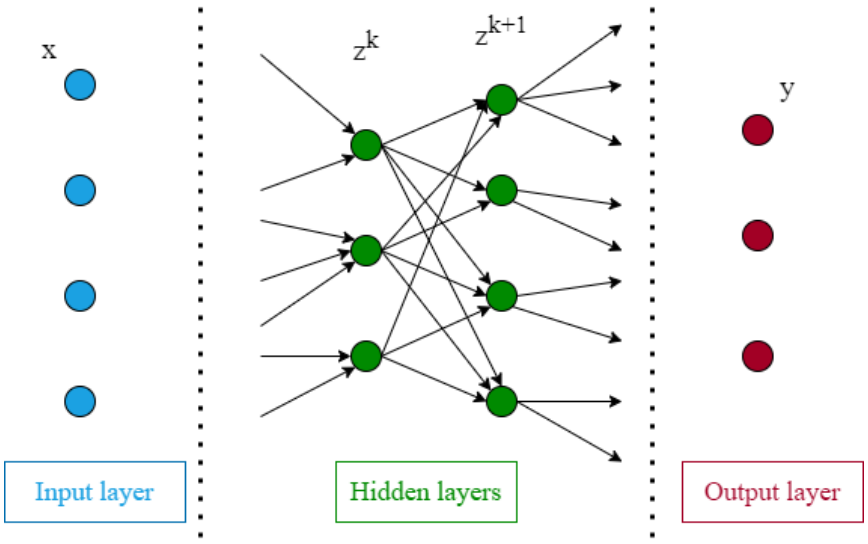


FIGURE 4.1: Illustration of a deep neural network

What makes the model (4.1) so powerful is that the way information is transferred from the input layer to the output layer can be arbitrarily complex thus it is possible to represent a very large set of classifier functions $\hat{c}(x)$ which can be very close to the true function $c(x)$. Typically, we set $x = z^0$ then we repeat for each layer k

$$z_i^{k+1} = h_k\Big(\sum_j w_{i,j}^k z_j^k\Big)$$

where $h_k : \mathbb{R} \to \mathbb{R}$ is a well-chosen nonlinear function and $w_{i,j}^k \in \mathbb{R}$ is the weight on the arc going from node j in the layer k to node i in the next layer. In general, the weights $w_{i,j}^k$ are "learned" by training the neural network on a dataset via gradient method done in a particular way. We refer to Papernot et al. [18] for more details.

4.2 Adversarial setup

As explained previously, the deep neural networks are often quite sensitive to slightly modified inputs called adversarial examples, see also Ughi, Abrol, and Tanner [20] and Ilyas et al. [14]. We talk about adversarial attack when we refer to the process for building an adversarial example. In order to find such inputs, we consider ourselves as an opponent of the neural network with the following aim :

Given a classifier $\hat{c} : \mathcal{X} \to \mathcal{C}$ and $x_0 \in \mathcal{X}$ correctly classified, we look for $x \in \mathcal{X}$ such that

$$\hat{c}(x) \neq \hat{c}(x_0) \quad \text{and} \quad ||x - x_0||_\infty \leq \delta$$

with the perturbation bound $\delta > 0$.

We consider perturbations that are bounded with respect to the infinite norm denoted by $||.||_\infty$. Typically δ is taken sufficiently small so that x actually belongs to the same class as x_0 so we have $c(x_0) = c(x) = \hat{c}(x_0)$. Therefore, if we are able to find such x not too far from x_0 then we know that x is misclassified by $\hat{c}$ hence our attack manage to mislead the classifier. The adversarial attacks can be separated into two main categories.

- ✧ **Non-targeted attack :** The adversary wants to modify x_0 in a way that it is classified in a class different from $c(x_0)$ without any preferences.
- ✧ **Targeted attack :** The adversary must specify beforehand a class $c_t \in \mathcal{C}$ with $c_t \neq c(x_0)$ then find a perturbed version of x_0 classified as c_t.

We can notice that targeted adversarial attacks are harder than non-targeted attacks because we add a constraint on the classification of the adversarial example. Non-targeted examples are useful when we are only interested in excluding one of the classes but we should pay attention when the classifier can classify the input in classes which are barely the same. Indeed, adversarial attack on an image classifier leading to $c(x) = "Llama"$ given that $c(x_0) = "Alpaca"$ may not be considered as fully successful as this two animals are already quite similar.

We will develop an algorithm able to generate targeted adversarial examples against neural networks. As said previously, the output for such classifiers is generally a conditional class probability $y_i(x) = P(c_i|x)$. Nevertheless, the amount of information available about $y(x)$ is very limited in practice, we distinguish

- ✧ **White-box setting :** The adversary has a full knowledge about the network architecture and the weights so it is possible to compute $y_i(x)$, $\nabla y_i(x)$ for all i.
- ✧ **Black-box setting :** The adversary can only ask to receive the class probability $y_i(x)$ for all i and the neural network is considered as hidden to the attacker.

White-box adversarial examples can be built easily via gradient-based method but having as much information about the classifier is quite uncommon. It only happens for open source softwares or attacks on neural networks trained by the attacker himself. In the real world, the commercial neural networks are black-boxes and even if we know on which data set the classifier was trained, it is usually impossible to know exactly how it was done.

Furthermore, we may be restricted on the maximal number of calls to $y(x)$. Because many classifiers either limit the number of queries with a threshold or still accept queries but with an additional cost for each new prediction. Consequently, the need for efficient algorithms in terms of function evaluations is not only motivated by the algorithm's running time but also by money.

4.3 Mathematical modeling

Analogously to the previous chapters, we consider iterative methods starting with the well-classified input x_0 and hope that at some point, an iterate x_k generated by the method will be an adversarial example. A way to create a targeted adversarial example is to solve the following constrained optimization problem.

$$\min_{x \in \mathcal{X}} L(x) \quad \text{such that} \quad ||x - x_0||_\infty \leq \delta \tag{4.2}$$

where $L : \mathcal{X} \to \mathbb{R}$ measures how "well" x is assigned to the targeted class c_t, for instance :

$$L(x) = -\ln y_t(x) + \sum_{i \neq t} \ln y_i(x)$$

$$\implies \quad \nabla L(x) = -\frac{1}{y_t(x)} \nabla y_t(x) + \sum_{i \neq t} \frac{1}{y_i(x)} \nabla y_i(x)$$

In the white-box setting, the problem (4.2) can be solved with a projected gradient method namely repeat for $k = 0, 1, 2, \ldots$ until x_k is misclasified

$$x_{k+1} = P_\delta(x_k - \alpha_k \nabla L(x_k))$$

where P_δ denotes the orthogonal projection on the set $\{x \in \mathcal{X} \mid ||x - x_0||_\infty \leq \delta\}$. However, if we are not allowed to compute $\nabla L(x)$ then the optimization problem (4.2) becomes harder.

A possible remedy is to create a substitute neural network trained with a data set obtained by querying the initial network. Afterwards, we can apply techniques for generating white-box adversarial examples on this new network instead of black-box attacks on the targeted network. Both neural networks might have different architectures but we expect that they will behave in a similar way namely the adversarial examples of the substitute classifier are likely to be adversarial example of the original classifier too, details on this approach can be found in Papernot et al. [18].

A drawback of the substitute neural network is that many predictions may be required for building it, especially if we are only interested in making few adversarial examples. Furthermore, it is not possible to guarantee that the perturbed inputs found with the white-box attack on the substitute network are truly adversarial examples.

Thereafter, we consider the next mathematical formulation for black-box adversarial attacks which can be solved via the methods seen in Chapter 2.

> Given the output layer $y : \mathcal{X} \to \mathbb{R}^m$ of a neural network and $x_0 \in \mathcal{X}$, we look for $x \in \mathcal{X}$ such that
>
> $$y(x) = y_{\text{target}} \quad \text{and} \quad ||x - x_0||_\infty \leq \delta$$
>
> with the targeted score $y_{\text{target}} \in \mathbb{R}^m$ and the perturbation bound $\delta > 0$.

Notice that $F(x) = y(x) - y_{\text{target}} = 0$ is an underdetermined system of nonlinear equations which must be solved with a derivative-free method. If y_{target} is chosen properly then an input satisfying the previous constraint is an adversarial example but this problem might seem harder as we enforce a constraint on the whole set of conditional probabilities instead of the final classification. Nevertheless, we don't really need to run the method until $y(x) = y_{\text{target}}$ but we can stop iterating when we found an adversarial example.

4.4 Practical implementation

We consider that the classes are numbered hence $\mathcal{C} = \{1, 2, \dots, m\}$. We would like to make an adversarial example x classified in the targeted class $t \in \mathcal{C}$. Nevertheless, the classifier has often quite accurate ideas about the category corresponding to the initial input x_0. Therefore, we need to act in such way that the perturbed input promotes the class t but also avoids (or favors) other classes in some set $\mathcal{A} \in \mathcal{C}$.

Remark that all the classes able to challenge the targeted classification of x must belong to $\mathcal{A}$. Although, classes might only take importance during the iterative process, making the choice of $\mathcal{A}$ beforehand sometimes quite challenging.

Given a target $t \in \mathcal{C}$, a set $\mathcal{A} \subset \mathcal{C}$ and $\xi \in (0, 1]$, we define $y_{\text{target}} \in \mathbb{R}^m$ as follow

$$(y_{\text{target}})_i = \begin{cases} \xi\, Y & \text{if} \quad i = t \\ \frac{1-\xi}{|\mathcal{A}|}\, Y & \text{if} \quad i \in \mathcal{A} \\ y_i(x_0) & \text{otherwise} \end{cases} \qquad \text{with} \quad Y = \sum_{j \in \{t\} \cup \mathcal{A}} y_j(x_0)$$

Notice that y_{target} represents a probability distribution because

$$\sum_{i=0}^{m} (y_{\text{target}})_i = 1 \quad \text{and} \quad (y_{\text{target}})_i \geq 0 \quad \text{for } i = 1, 2, \dots, m$$

which is required for feasibility. Intuitively, we look at the aggregated probability Y that x_0 belong to a class in $\{t\} \cup \mathcal{A}$ and after we redistribute it between t and $\mathcal{A}$. The parameter ξ controls the weight given to the targeted class meaning that ξ close to 1 gives almost zero probabilities to avoided classes while $\xi = \frac{1}{2}$ tends to be more fair between t and $\mathcal{A}$. Sometimes, we would like to allow other classes than the target to have a potentially non negligible probability in order to weaken the initial class of x_0 and obtain more easily a misclassification.

To summarize what was said before, we want to find $x \in \mathbb{R}^n$ such that

$$F(x) = y(x) - y_{\text{target}} = 0 \tag{4.3}$$
$$||x - x_0||_\infty \leq \delta \tag{4.4}$$

We have already methods that can solve equation (4.3). However, we need to do small adjustments to *Algorithm 8* so that it can also handle the constraint (4.4). Typically, this can be done by using a modified line search condition given by

$$f(x_{k+1}) \leq f(x_k) + \nu_k + \theta_k - \rho\gamma_k(\alpha_k\beta^l)^2 f(x_k) \quad \text{and} \quad ||x_{k+1} - x_0||_\infty \leq \delta$$

Moreover, we use the random definition of the matrices $(E_k)_{k=0}^{\infty}$ because the components in $F(x)$ with the highest magnitude are usually always the same hence we don't want to have a periodic sequence. Indeed, a periodic sequence in the case of image classification would mean that only few pixels change the most. For reproducibility, we fixed the seed in MATLAB as follow

```
rng(0,'twister')
```

4.5 Black-box attack on googlenet

We experiment our techniques to generate adversarial examples on the image classifier googlenet available at https://nl.mathworks.com/help/deeplearning/ug/classify-image-using-googlenet.html. Googlenet is a deep neural network trained on the ImageNet database and made of 22 layers. The network takes as input an image with size 224×224 under RGB representation meaning that each pixel is represented by three components of color red, green and blue.

For an image x, googlenet returns the vector of conditional probabilities $y(x)$ for 1000 possible categories such as *"espressomaker"*, *"ski"*, *"minivan"* and also many kind of animals. In the next experiments, we use a perturbation bound $\delta = 0.5$ and the parameters of *Algorithm 8* are the same as in the *Chapter 3*, except the precision which is lowered to $\epsilon = 10^{-2}$.

In addition, we change the stopping criterion and consider that we found a good adversarial example when the probability of the target $t \in \mathcal{C}$ satisfies

$$y_t(x) \geq \mu \max_{i \in \mathcal{C},\, i \neq t} y_i(x) \tag{4.5}$$

If we pick $\mu = 1$ then criterion (4.5) is equivalent to stopping the method when we found an adversarial example. However, the probability $y_t(x)$ might be very close to certain other class probabilities. Thus, we add the parameter $\mu \geq 0$ to say that the targeted class must be at least μ times more likely than any other classes. All the next experiments are done with the value $\mu = 1.1$.

It is worth to mention that we keep in memory the image with the larger value of $y_t(x)$ during the iteration process. It means that the targeted class probability is always increasing.

We choose to focus mainly on the Memoryless method as it seems to be the fastest when it converges and the comparison of the methods with the same value of ξ is not always adequate. Additional results for DF-SAUNE can be found in *Appendix B*. We don't provide an analysis of the Adaptive method because it struggles a bit with underdetermined systems of equations, see *Chapter 3*.

❒ **First adversarial example**

We start with the image of a lesser panda given by *Figure 4.2*. This image is correctly classified with probability 0.973 which is actually quite high. Despite this, the classifier attributes a significant probability to the red fox as they are similar animals, the remaining classes are considered as very unlikely. Based on this information, we will manage to find an adversarial example classified as $t = "Redfox"$ and the set of avoided classes in this case will simply be $\mathcal{A} = \{"Lesser\ panda"\}$.

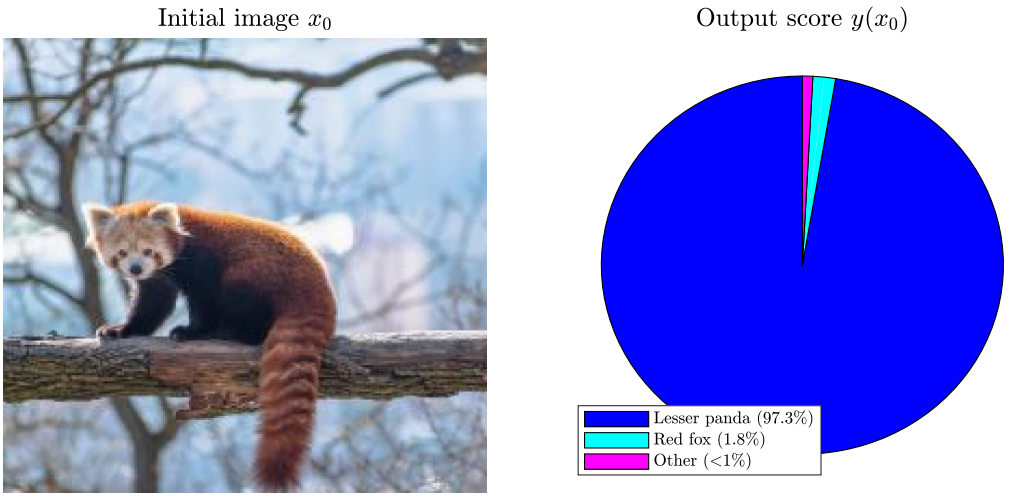


FIGURE 4.2: Initial image representing a lesser panda [1]

For this example, we fix $\xi = 0.65$. Other choices are possible but large values might lead to an infeasible problem. Indeed, if the targeted probability of the class t is too high then it can be impossible to reach it because of the perturbation bound. The value of ξ is found by testing different guesses and then picking the best, see *Figure B.1* and *Figure B.2* in *Appendix B*.

We observe in *Figure 4.3* that the sum of both probabilities is always close to 1 meaning that we don't have issues with classes outside of $\mathcal{A} \cup \{t\}$. Therefore, we focus on the required number of function evaluations to observe the intersection between the curves *"Lesser panda"* and *"Red fox"* which will correspond to an adversarial example. We notice that the Memoryless method needs needs approximately 6500 function evaluations to get an adversarial example.

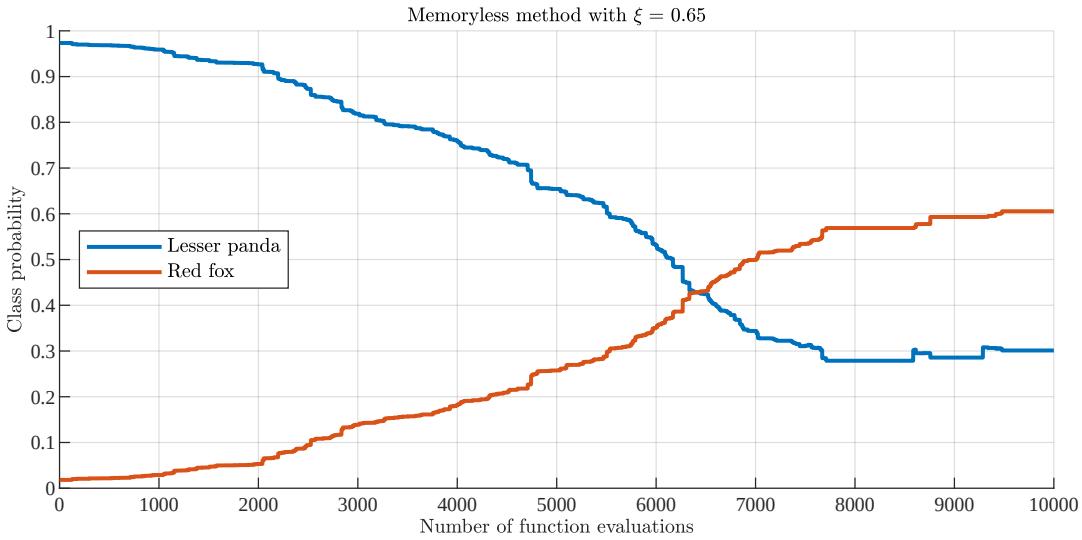


FIGURE 4.3: Adversarial attack via Memoryless method for *Figure 4.2*

[1] Image available here : https://www.animalwised.com/why-is-the-red-panda-endangered-4097.html

The adversarial example found by the Memoryless method with the stopping criterion (4.5) is shown in *Figure 4.4*. The method used 6560 function evaluations to find this adversarial example which looks a bit like *Figure 4.2*. However, if we have a closer look then we can see few pixels with a modified color, see *Figure 4.5*. Notice that this adversarial example doesn't really solve $y(x) = y_{\text{target}}$ according to *Figure 4.4* because as we increase the probability of *"Red fox"*, other species of fox appear while they should have a nearly zero probability in y_{target}. Therefore, we might think about adding other classes to $\mathcal{A}$ in order to make the problem "more feasible".

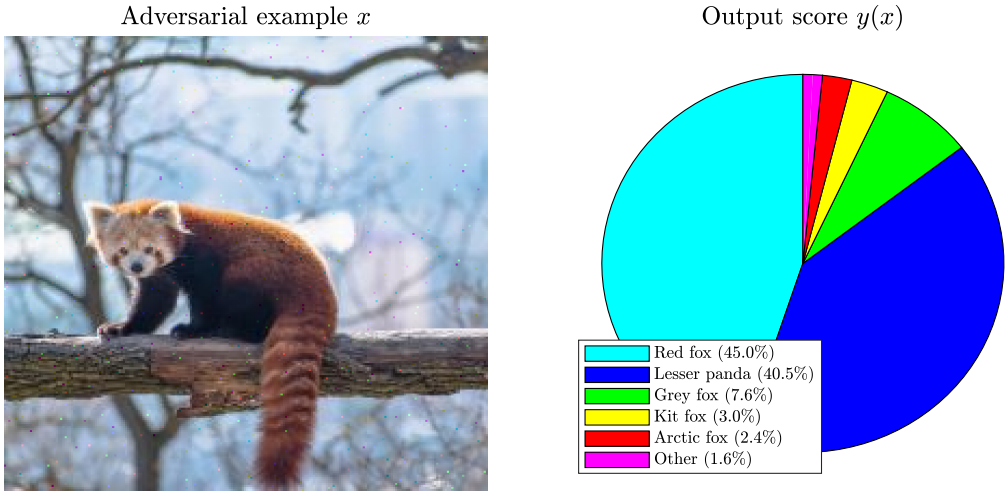


FIGURE 4.4: Adversarial example *"Lesser panda"* found with the Memoryless method, $\xi = 0.65$ and $\mathcal{A} = \{$*"Lesser panda"*$\}$

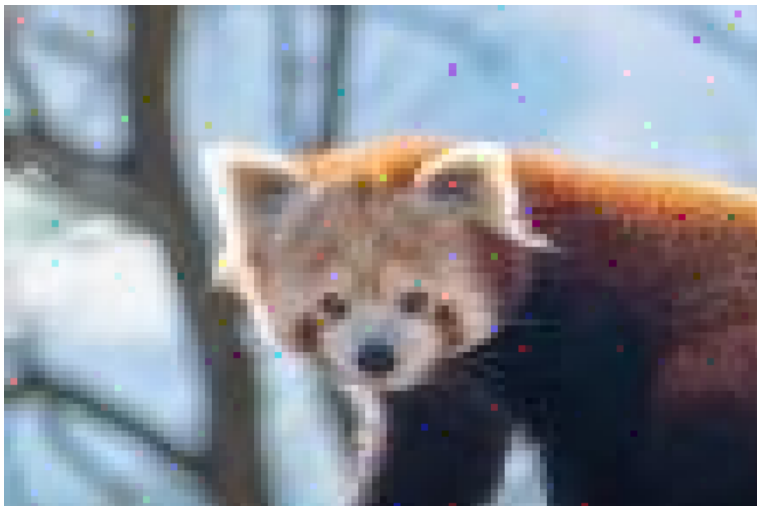


FIGURE 4.5: Closer look on the adversarial example *"Lesser panda"*

❒ Second adversarial example

Figure 4.6 is a picture showing a llama which is clearly well-identified by the classifier then follow in order of probability the animals *"Ram"*, *"Bighorn"* (two kind of sheeps), *"Kuvasz"* (a species of dog) and *"White wolf"*. We will aim to classify this llama as a white wolf. It is a bit harder than the previous case because there is more than one class which is more likely than the target. In addition, the initial probability is lower 0.004 instead of 0.018 before with the red fox.

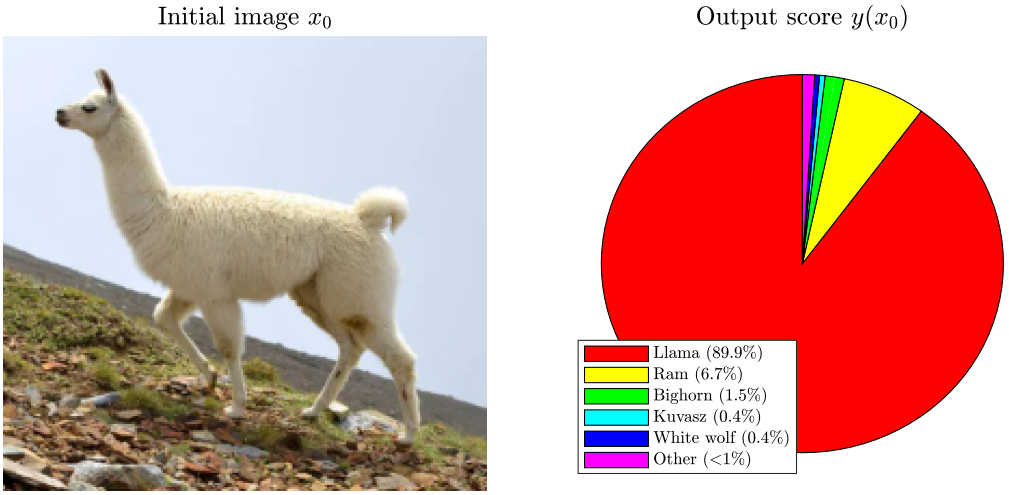


FIGURE 4.6: Initial image representing a llama [2]

We consider the same greedy approach as the previous attack meaning that we choose $t = "White\ wolf"$ and $\mathcal{A} = \{"Llama"\}$. However, the first result with $\xi = 0.65$ presented in *Figure 4.7*, is a little bit disappointing since the method gets quickly stuck and can't do better than a probability of 0.52 for the llama and 0.1 for the white wolf. It means that the probability y_{target} is too difficult to reach. Furthermore, it seems quite hard to find good values of ξ which would allow us to build an adversarial example, see *Figure B.3* and *Figure B.4* in *Appendix B*.

[2]Image available here : https://fr.depositphotos.com/stock-photos/llama.html

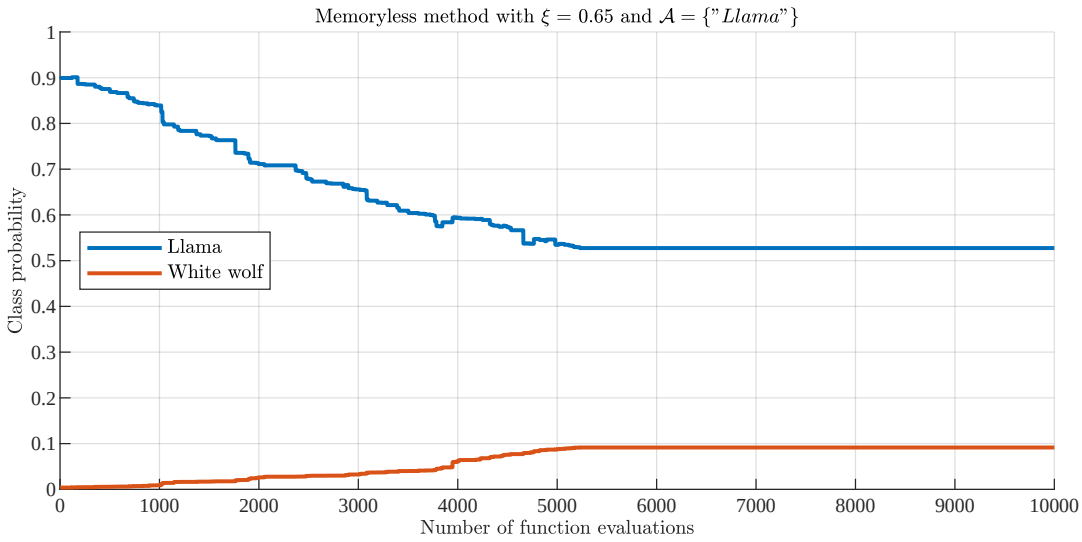


FIGURE 4.7: Adversarial attack via Memoryless method for *Figure 4.6*

Therefore, we need to make a guess based on the output score of the initial image by taking some inspiration from the last example with the foxes. We assume that if the perturbed image looks like a white wolf then it should also looks like a kuvasz as it is a kind of white dog. Consequently, allowing an increase of the kuvasz probability should make the problem feasible hence we define $\mathcal{A} = \{"Llama", "Kuvasz"\}$.

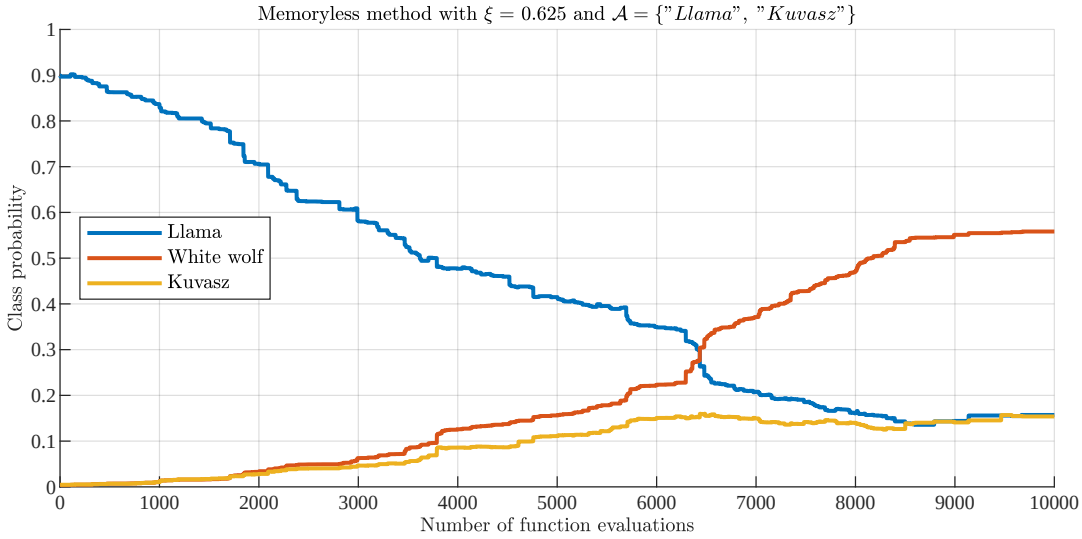
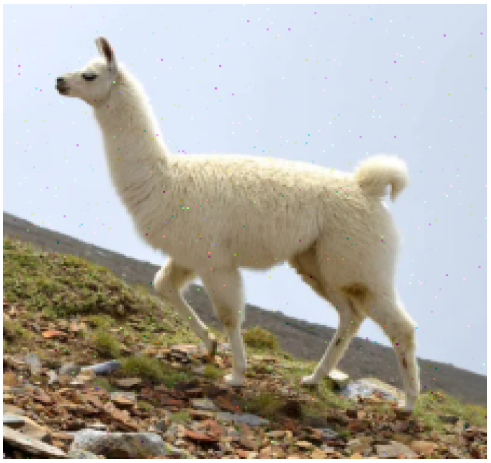


FIGURE 4.8: Adversarial attack via Memoryless method for *Figure 4.6*

With this approach, it is not too difficult to find a targeted distribution which works, for instance the case $\xi = 0.625$ is shown in *Figure 4.8*. We observe that both the probabilities of *"Kuvasz"* and *"White wolf"* increase at the beginning. Afterwards, the *"White wolf"* becomes more likely than *"Llama"* and the probability of *"Kuvasz"* at the end is nearly equal to the one of *"Llama"* which makes sense has all the classes in $\mathcal{A}$ have the same targeted probability. The resulting adversarial example found with 6435 function evaluations is given in *Figure 4.9*.

Adversarial example x



Output score $y(x)$

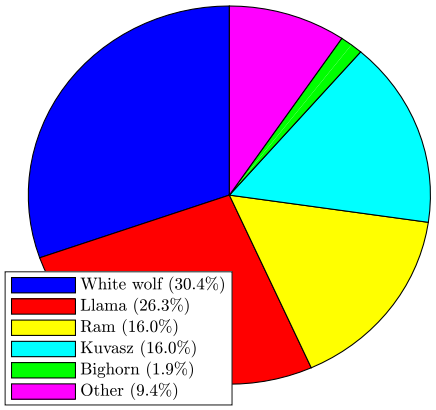


FIGURE 4.9: Adversarial example "*Llama*" found with the Memoryless method, $\xi = 0.625$ and $\mathcal{A} = \{"Llama", "Kuvasz"\}$

In conclusion, an advantage of this technique compared to derivative-free optimization methods, is that each class probability can be specified, instead of only minimizing one class probability. We think that the most challenging part about this technique is to find a relatively good target distribution y_{target}. Nevertheless, when this is done, the Memoryless method is able to find an adversarial example with approximately 6000-7000 queries to the neural network.

Chapter 5

Conclusion

To sum up, the main aim of this work was to analyze new derivative-free methods for solving underdetermined systems of nonlinear equations. The analysis focused both on theoretical results and numerical experiments. On the top of that, we also described a new technique based on these derivative-free methods allowing to build targeted adversarial examples in the black-box setting.

Firstly, we presented a new class of methods generalizing the work from Grapiglia and Chorobura [11]. This set of new methods uses a modified spectral search direction like Echebest, Schuverdt, and Vignau [9] to handle underdetermined systems of equations. Concerning the search direction $d_k = -\sigma_k E_k^T F(x_k)$, a larger set of matrices E_k was considered in our analysis as the only property required in the proofs was $E_k E_k^T = I$. We showed that there were many degrees of freedom in this new class of methods. Indeed, the sequences $(\sigma_k)_{k=0}^{\infty}$, $(\theta_k)_{k=0}^{\infty}$, $(\nu_k)_{k=0}^{\infty}$, $(E_k)_{k=0}^{\infty}$ and $(\alpha_k)_{k=0}^{\infty}$ can be defined arbitrarily. It was only required that $\sigma_{\min} \leq |\sigma_k| \leq \sigma_{\max}$, $\nu_k \geq 0$, $\theta_k > 0$ and $\sum_{k=0}^{\infty} \theta_k < \infty$. Afterwards, we focused on two new methods, the Memoryless method and Adaptive method, differing only by the choice for $(\alpha_k)_{k=0}^{\infty}$. From the theoretical view point, we managed to compute the maximal number of function evaluations required to generate an iterate x_k such that $\psi_k(x_k) \leq \epsilon$ with $\epsilon \in (0,1)$. We founded complexity bounds in $\mathcal{O}\left(\epsilon^{-2}\log(\epsilon^{-1})\right)$ and $\mathcal{O}\left(\epsilon^{-4}\right)$ function evaluations respectively for the Memoryless method and the Adaptive method.

Secondly, three methods belonging to the new class mentioned above, specifically DF-SAUNE, the Memoryless method and the Adaptive method were tested. In addition, we compared them via data profiles, a tool from Moré and Wild [17]. Despite its worst theoretical complexity bound, we observed that the Adaptive method was the most competitive method for solving determined systems of equations. However, the Adapative method was usually less efficient than DF-SAUNE and the Memoryless method while tested on underdetermined systems of equations. It may be due to the fact that the change of matrix E_k at each step deteriorates the information provided by the last step. Finally, we did few experiments with a new sequence $(E_k)_{k=0}^{\infty}$ and showed that the relation between m and n can have a huge effect on the behavior of the methods.

Thirdly, we modeled the creation of adversarial attacks as an underdetermined system of equations $y(x) = y_{\text{target}}$ with $||x - x_0||_\infty \leq \delta$. In order to solve this problem, the new class of methods was modified by adding the bounded perturbation constraint to the line search condition. Moreover, we use the random strategy for $(E_k)_{k=0}^{\infty}$ to avoid prioritizing only few pixels. It appeared that the more challenging part was to find a "good" targeted class probability y_{target}. We also remarked

that if we manage to find such probability then our technique could build relatively quickly an adversarial example.

Finally, the next suggested improvements about this class of methods should be done on the way the sequences $(\nu_k)_{k=0}^{\infty}$, $(E_k)_{k=0}^{\infty}$ and $(\alpha_k)_{k=0}^{\infty}$ are defined. Indeed, in the proofs about the complexity bounds, we required $\sum_{k=0}^{\infty} \nu_k < \infty$ but this somehow restricts a lot the nonmonotonicity. Thus, it might be interesting to see if alternative proofs avoiding such constraint, can demonstrate similar complexity bounds for particular choices of ν_k. Moreover, selecting E_k either heuristically or based on the values m and n may lead to improved methods. Finally, in the case of sequence $(E_k)_{k=0}^{\infty}$ with period N, one could think about modifying the Adaptive method so that the guess α_k is based on the information from the step $k - N$ instead of $k - 1$ or even several steps. An other direction of research for the black-box adversarial attacks could be to automate the process of defining y_{target} such as a more efficient trial and error process or accurate guesses based on the initial image.

Appendix A

Test references

Problem	m	n	Source
1	2	2	Problem 1 in Moré, Garbow, and Hillstrom [16]
2	2	2	Problem 2 in Moré, Garbow, and Hillstrom [16]
3	2	2	Problem 3 in Moré, Garbow, and Hillstrom [16]
4	3	3	Problem 7 in Moré, Garbow, and Hillstrom [16]
5	4	4	Problem 13 in Moré, Garbow, and Hillstrom [16]
6	10	10	Problem 21 in Moré, Garbow, and Hillstrom [16]
7	12	12	Problem 22 in Moré, Garbow, and Hillstrom [16]
8	10	10	Problem 26 in Moré, Garbow, and Hillstrom [16]
9	40	40	Problem 27 in Moré, Garbow, and Hillstrom [16]
10	10	10	Problem 28 in Moré, Garbow, and Hillstrom [16]
11	10	10	Problem 29 in Moré, Garbow, and Hillstrom [16]
12	10	10	Problem 30 in Moré, Garbow, and Hillstrom [16]
13	10	10	Problem 31 in Moré, Garbow, and Hillstrom [16]
14	10	10	Problem 32 in Moré, Garbow, and Hillstrom [16]
15	10	10	Problem 33 in Moré, Garbow, and Hillstrom [16]
16	10	10	Problem 34 in Moré, Garbow, and Hillstrom [16]
17	1	2	Problem 6 in Hock and Schittkowski [13]
18	1	2	Problem 7 in Hock and Schittkowski [13]
19	2	2	Problem 8 in Hock and Schittkowski [13]
20	1	3	Problem 26 in Hock and Schittkowski [13]
21	1	3	Problem 27 in Hock and Schittkowski [13]
22	2	4	Problem 39 in Hock and Schittkowski [13]
23	3	4	Problem 40 in Hock and Schittkowski [13]
24	2	4	Problem 42 in Hock and Schittkowski [13]
25	2	5	Problem 46 in Hock and Schittkowski [13]
26	3	5	Problem 47 in Hock and Schittkowski [13]
27	2	5	Problem 48 in Hock and Schittkowski [13]
28	3	5	Problem 53 in Hock and Schittkowski [13]
29	4	5	Problem 56 in Hock and Schittkowski [13]
30	2	7	Problem 61 in Hock and Schittkowski [13]
30	2	3	Problem 61 in Hock and Schittkowski [13]
31	2	3	Problem 63 in Hock and Schittkowski [13]
32	2	5	Problem 77 in Hock and Schittkowski [13]
33	3	5	Problem 78 in Hock and Schittkowski [13]
34	3	5	Problem 79 in Hock and Schittkowski [13]
35	3	5	Problem 81 in Hock and Schittkowski [13]
36	3	10	Problem 111 in Hock and Schittkowski [13]
37	300	300	Problem 2 a) in Dan, Yamashita, and Fukushima [7]
38	300	300	Problem 2 b) in Dan, Yamashita, and Fukushima [7]
39	150	300	Problem 4 a) in Dan, Yamashita, and Fukushima [7]
40	150	300	Problem 4 b) in Dan, Yamashita, and Fukushima [7]
41	149	150	Problem 42 in Zaslavski and Powell [22]
42	299	300	Problem 42 in Zaslavski and Powell [22]

Appendix B

Additional results about adversarial attacks

✧ **First adversarial example :** *"Lesser panda"*

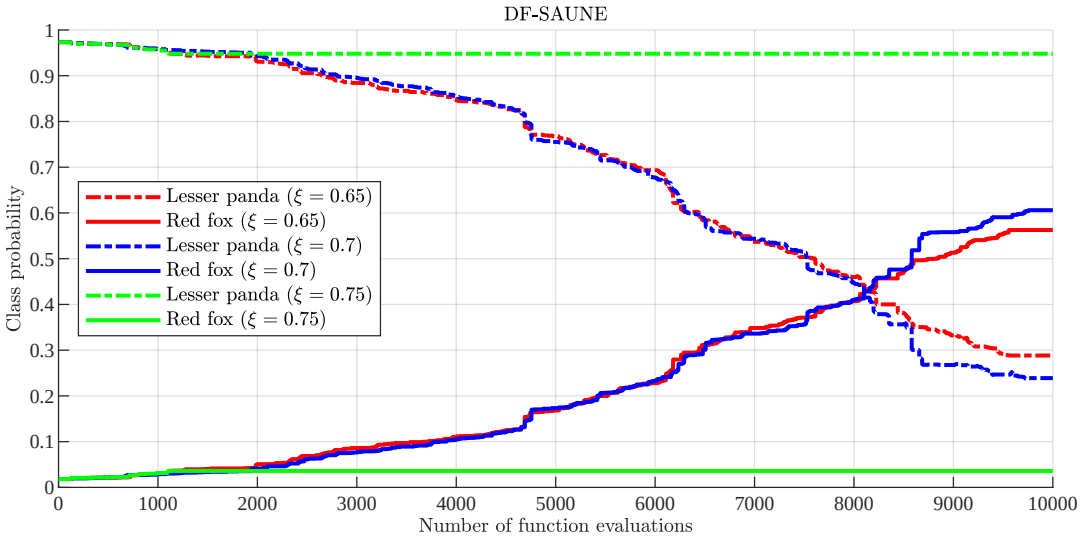


FIGURE B.1: Adversarial attack via DF-SAUNE for *Figure 4.2*

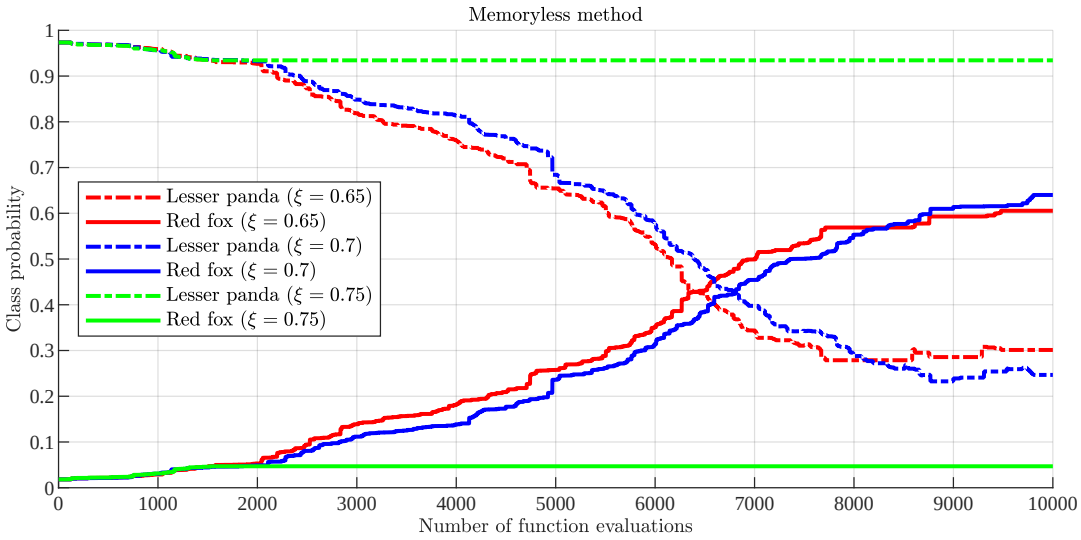


FIGURE B.2: Adversarial attack via Memoryless method for *Figure 4.2*

✧ **Second adversarial example :** *"Llama"*

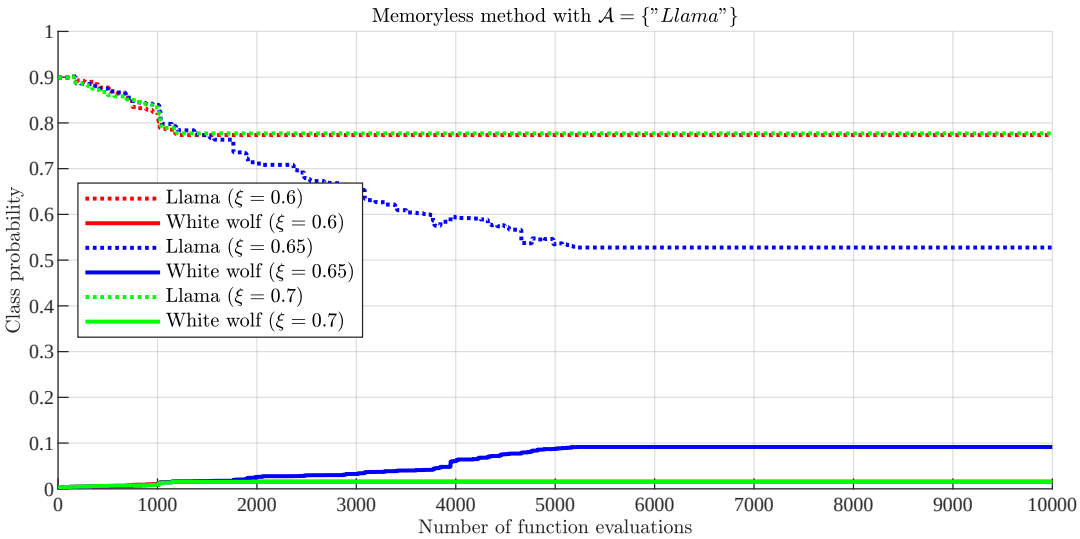


FIGURE B.3: First attempt via Memoryless method for *Figure 4.6*

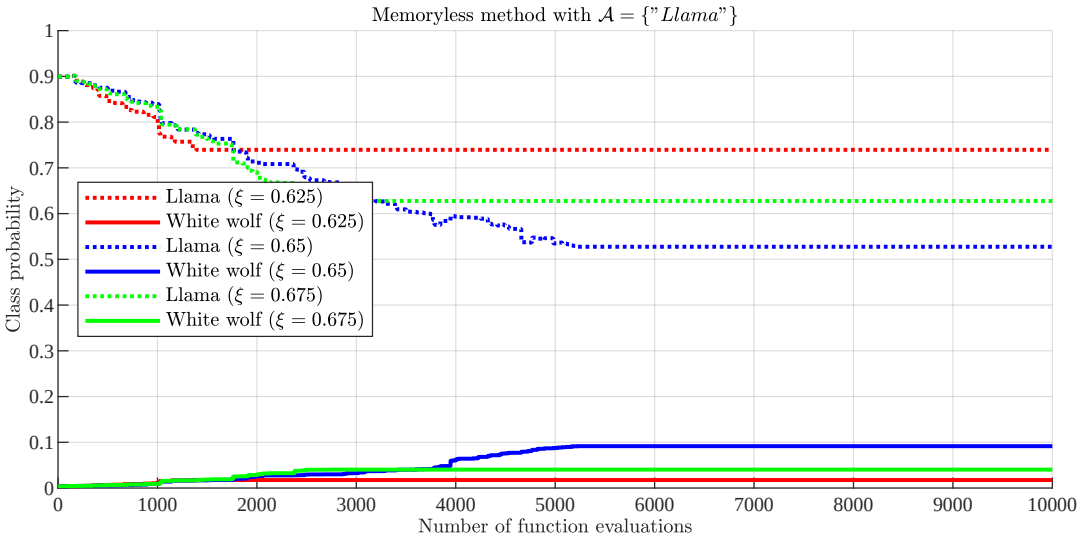


FIGURE B.4: Second attempt via Memoryless method for *Figure 4.6*

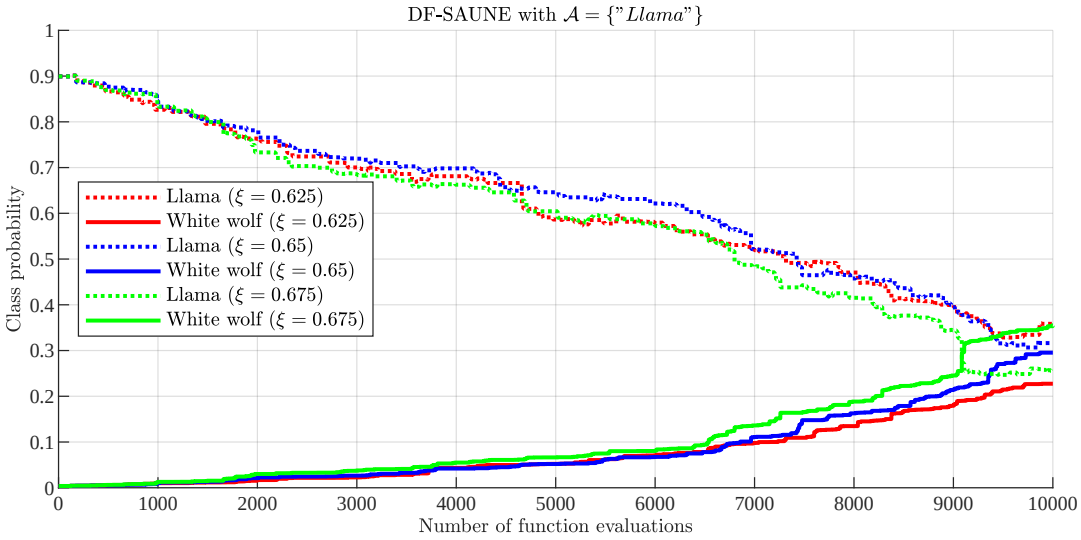


FIGURE B.5: Adversarial attack via DF-SAUNE for *Figure 4.6*

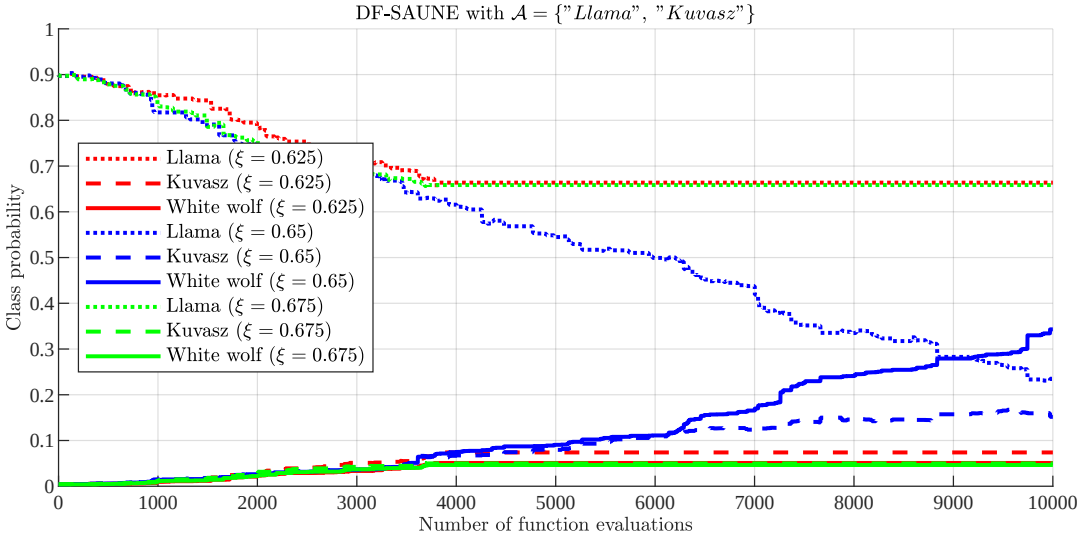


FIGURE B.6: Adversarial attack via DF-SAUNE for *Figure 4.6*

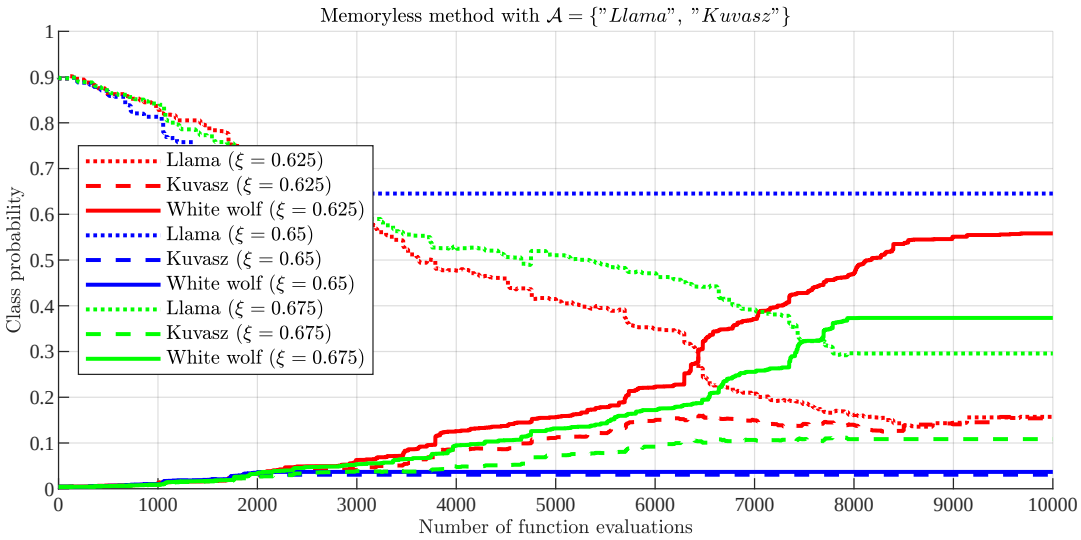


FIGURE B.7: Adversarial attack via Memoryless method for *Figure 4.6*

Bibliography

[1] Larry Armijo. "Minimization of functions having Lipschitz continuous first partial derivatives". In: *Pacific Journal of mathematics* 16.1 (1966), pp. 1–3. DOI: 10.2140/pjm.1966.16.1.

[2] Jonathan Barzilai and Jonathan M. Borwein. "Two-Point Step Size Gradient Methods". In: *IMA Journal of Numerical Analysis* 8.1 (1988), pp. 141–148. DOI: 10.1093/imanum/8.1.141.

[3] E. G. Birgin, J. M. Martínez, and M. Raydan. "Spectral Projected Gradient methods: Review and Perspectives". In: *Journal of Statistical Software* 60.3 (2014), pp. 1–21. DOI: 10.18637/jss.v060.i03.

[4] C. G. Broyden. "Newton's method and its use in optimization". In: *Mathematics of Computation* 19.92 (1965), 577–93. DOI: 10.2307/2003941.

[5] Wanyou Cheng and Dong-Hui Li. "A derivative-free nonmonotone line search and its application to the spectral residual method". In: *IMA Journal of Numerical Analysis* 29 (July 2009). DOI: 10.1093/imanum/drn019.

[6] William La Cruz, José Mario Martínez, and Marcos Raydan. "Spectral residual method without gradient information for solving large-scale nonlinear systems of equations". In: *Mathematics of Computation* 75.255 (2006), pp. 1429–1448. DOI: 10.1090/S0025-5718-06-01840-0.

[7] Hiroshige Dan, Nobuo Yamashita, and Masao Fukushima. "Convergence Properties of the Inexact Levenberg-Marquardt Method under Local Error Bound Conditions". In: (2010). DOI: 10.1080/10556780210000049345.

[8] J. E. Dennis Jr. and Jorge J. Moré. "Quasi-Newton Methods, Motivation and Theory". In: *SIAM Review* 19.1 (1977), pp. 46–89. DOI: 10.1137/1019005.

[9] N. Echebest, M.L. Schuverdt, and R.P. Vignau. "Two derivative-free methods for solving underdetermined nonlinear systems of equations". In: *Computational and Applied Mathematics* 30.1 (2011), 217–245. DOI: 10.1016/j.amc.2012.09.056.

[10] Carl Geiger and Christian Kanzow. "On the resolution of monotone complementarity problems". In: *Computational Optimization and Applications* 5.1 (1996), pp. 155–173. DOI: 10.1007/BF00249054.

[11] Geovani N. Grapiglia and Flávia Chorobura. "Worst-case evaluation complexity of derivative-free nonmonotone line search methods for solving nonlinear systems of equations". In: *Computational and Applied Mathematics* 40.259 (2021). DOI: 10.1007/s40314-021-01621-4.

[12] L. Grippo, F. Lampariello, and S. Lucidi. "A nonmonotone line search technique for Newton's method underdetermined nonlinear systems of equations". In: *SIAM J. on Numerical Analysis* 23 (1986), 707–716. DOI: 10.1137/0723046.

[13] Willi Hock and Klaus Schittkowski. "Test Examples for Nonlinear Programming Codes". In: (1981). DOI: 10.1007/978-3-642-48320-2.

[14] Andrew Ilyas et al. "Black-box Adversarial Attacks with Limited Queries and Information". In: *Proceedings of the 35th International Conference on Machine Learning* 80 (2018). DOI: 10.48550/arXiv.1804.08598.

[15] D.H. Li and M. Fukushima. "A derivative-free line search and global convergence of Broyden-like method for nonlinear equations". In: *Opt. Meth. and Software* 13 (2000), 181–201. DOI: 10.1080/10556780008805782.

[16] Jorge J. Moré, Burton S. Garbow, and Kenneth E. Hillstrom. "Testing Unconstrained Optimization Software". In: *ACM Trans. Math. Softw.* 7.1 (1981), 17–41. ISSN: 0098-3500. DOI: 10.1145/355934.355936.

[17] Jorge Moré and Stefan Wild. "Benchmarking Derivative-Free Optimization Algorithms". In: *SIAM Journal on Optimization* 20 (Jan. 2009), pp. 172–191. DOI: 10.1137/080724083.

[18] Nicolas Papernot et al. "Practical Black-Box Attacks against Machine Learning". In: (2017), 506–519. DOI: 10.1145/3052973.3053009.

[19] Boris Polyak. "Newton's method and its use in optimization". In: *European Journal of Operational Research* 181 (Sept. 2007), pp. 1086–1096. DOI: 10.1016/j.ejor.2005.06.076.

[20] Giuseppe Ughi, Vinayak Abrol, and Jared Tanner. "An empirical study of derivative-free optimization algorithms for targeted black-box attacks in deep neural networks". In: *Optimization and Engineering* 23 (2021), 1319–1346. DOI: 10.1007/s11081-021-09652-w.

[21] Homer F. Walker and Layne T. Watson. "Least-Change Secant Update Methods for Underdetermined Systems". In: *SIAM Journal on Numerical Analysis* 27.5 (1990), 1227–1262. DOI: 10.1137/0727071.

[22] Alexander Zaslavski and M. Powell. "The NEWUOA software for unconstrained optimization without derivatives". In: (Jan. 2006), pp. 255–297. DOI: 10.1007/0-387-30065-1_16.

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