SAPIENZA
UNIVERSITÀ DI ROMA

Enabling gradient-based optimization methods in problems with unreliable or absent derivatives

Department of Computer, Control and Management Engineering Antonio Ruberti

Dottorato di Ricerca in Automatica, Bioingegneria e Ricerca Operativa – XXXIV Ciclo

Candidate

Marco Boresta

ID number 1608203

Thesis Advisor

Prof. Stefano Lucidi

Co-Advisor

Prof. Alberto De Santis

20 May 2022

Thesis defended on 20 May 2022
in front of a Board of Examiners composed by:

Prof. Marco Sciandrone (chairman)

Prof. Costanzo Manes

Prof. Fabio Schoen

Enabling gradient-based optimization methods in problems with unreliable or absent derivatives
Ph.D. thesis. Sapienza – University of Rome

This thesis has been typeset by LaTeX and the Sapthesis class.

Author's email: marco.boresta@uniroma1.it

Contents

Chapter 1

Introduction

Many problems that arise in the fields of engineering, economics, and natural sciences can be represented as nonlinear optimization problems. This drives an ever-increasing technical and scientific interest in the study and development of methods capable of tackling and resolving this class of difficult mathematical problems. Among the various types of nonlinear optimization problems, we consider the general optimization problem with this structure:

$$\min_{x \in S} f(x) \tag{1.1}$$

where $S \subseteq \mathbb{R}^n$ and $f : \mathbb{R}^n \mapsto \mathbb{R}$.

In this thesis, we focus on problems in which the derivative of the objective function is either unavailable or unreliable, which can occur in a variety of situations including the presence of legacy codes (codes written in the past but not maintained), problems of parameter tuning for simulation or optimization algorithms and engineering problems where the objective functions are the output of black-box simulation software.

Despite the absence or the unreliability of the derivatives, our interest is in the resolution of the optimization problem using gradient-based methods, which take advantage of the rich and relevant information normally included in the gradient of the objective function.

We address the lack of derivatives considering two different scenarios. In the first one, we consider smooth problems with additive noise affecting objective function evaluations. We assume that objective function evaluations can be obtained in a cheap and fast way and we focus on gradient approximation methods that use objective function evaluations to somehow filter the noise and build an estimate of the gradient.

In the second scenario, we consider potentially non-smooth simulation-based optimization problems in which neither the objective function nor its (eventual) derivative have an explicit expression. Assuming the expensiveness of the evaluations of objective functions, we enable the usage of gradient-based methods by following an approach that is based on the creation of a neural network model that replaces

the simulation software used for computing the objective function. In this way, the smooth function obtained with the neural network model and its gradient are considered in the optimization procedure.

The thesis is structured as follows. In Chapter 2 we report some well known concepts about unconstrained optimization algorithms, with a special focus on gradient-based methods that are used in the numerical experiments of subsequent chapters.

In Chapter 3 we introduce the concept of function evaluations affected by noise and provide an overview of the most popular gradient approximation algorithms, showing their structure and some of their theoretical properties. We also report some known concepts about neural networks and their role as universal approximators.

In Chapter 4 we present the analysis that led to the development of a novel scheme for approximating the gradient of a function, that we name *Normalized Mixed Finite Difference Scheme*, or *NMXFD* for short. After providing its description, we present its theoretical properties and report two sets of numerical experiments in which we compare it to other approximation methods. In the first one, we perform a point-wise comparison between the gradient estimates produced by *NMXFD* and the ones obtained with other methods. The impact of different gradient approximation estimates on the performance of gradient-based optimization algorithms is investigated in the second set of experiments. A synthesis of the results of this chapter is published in [21].

Finally, in Chapter 5 we consider a case study concerning the operation of the Emergency Department of a large Italian hospital in Rome, with a focus on the Major Injury Units optimal resource allocation problem. This problem is modelled as a multi-objective simulation-based optimization problem that we reduce to a single objective one with a scalarization approach. To use gradient-based methods for its resolution, we propose a strategy based on the creation of a neural network model that replaces the simulation-based objective function and compare its performance with the ones obtained with a *Derivative-Free* approach. This chapter is based on a paper submitted by the author for publication and co-authored by Tommaso Giovannelli, Massimo Roma and Stefano Lucidi.

Chapter 2

Gradient methods for unconstrained optimization problems

In this chapter, we report some well known concepts about unconstrained optimization algorithms. In particular, we focus on gradient-based methods, showing their convergence properties and highlighting the importance of the information contained in the gradient. The algorithms, theorems and propositions presented in this chapter are well known. Readers interested in a more detailed analysis can refer to [20, 57, 99], and the references therein. The chapter is structured as follows: in Sect. 1 we introduce the general unconstrained optimization problem and provide a broad classification of solving algorithms. In Sect. 2 we focus on the Gradient Descent method and report its convergence properties. The analysis of the convergence rate in the case of quadratic functions considering different choices of the stepsize is the argument of Sect. 3. Finally, in Sect. 4 we consider the general case of non-quadratic functions and we report a nonmonotone algorithm which is based on the Barzilai-Borwein (BB) method and is proven to be globally convergent.

2.1 Generalities

Let us consider the following unconstrained optimization problem:

$$\min_{x \in \mathbb{R}^n} f(x) \tag{2.1}$$

where $f : \mathbb{R}^n \mapsto \mathbb{R}$ is a function with continuous derivative, i.e. $f \in \mathcal{C}^1(\mathbb{R}^n)$. Generally speaking, optimization algorithms can be classified both on the base of the characteristics of the problem they attempt to solve – i.e. depending on the objective function, the feasible set and the features of the variable space we can have *continous* or *integer* problems, *constrained* or *unconstrained* problems, *differentiable* and *non-differentiable* problems – and on the set of points that they accept as

solutions, denoted by Ω.
In particular, we can distinguish between *global optimization algorithms* and *local optimization algorithms*. The former try to tackle one of the hardest problems in the optimization field, since their set Ω is constituted by all the global minima of problem (2.1). The latter, produce a sequence of points $\{x^k\}$ satisfying some convergence properties towards local minima (or, more commonly in practice, stationary points) of the problem.

In Chapter 3 and 4 of this thesis, we focus on continuously differentiable problems in which the gradient of the objective function f exists. We denote it by $\nabla f : \mathbb{R}^n \mapsto \mathbb{R}^n$ such that, for any $x \in \mathbb{R}^n$:

$$\nabla f(x) = \begin{bmatrix} \frac{\partial f}{\partial x_1}(x) \\ \vdots \\ \frac{\partial f}{\partial x_n}(x) \end{bmatrix}. \tag{2.2}$$

We also focus on local optimization algorithms whose general scheme is reported in Algorithm 1.

Algorithm 1 General algorithm for unconstrained optimization

1: Choose $x^0 \in \mathbb{R}^n$, set $k = 0$
2: **while** $\nabla f(x^k) \neq 0$ **do**
3: Compute $s^k \in \mathbb{R}^n$
4: Set $x^{k+1} = x^k + s^k$
5: Set $k = k + 1$
6: **end while**

These methods try to use the local information that can be extracted from the problem (e.g., evaluating the objective function in points close to the current point x^k, computing the first or second derivatives of the objective function) and they try to exploit the fact that any function can be locally approximated pretty well by a linear or quadratic function.

In absence of specific properties of the objective function, the sequences of points generated by these methods have to converge, in some sense, towards points that satisfy the first order optimality conditions and whose value of the objective function is smaller or equal than the one at the starting point, as in

$$\Omega = \{x^* \in \mathbb{R}^n : \quad f(x^*) \leq f(x^0), \quad \nabla f(x^*) = 0\} \tag{2.3}$$

2.1.1 Convergence properties of local optimization algorithms

We can classify local algorithms based on their convergence properties, both in terms of reliability and efficiency. The reliability is normally linked to the concept of *global convergence* of the algorithm, while the efficiency to the concept of *convergence rate*.

Global convergence

Globally convergent algorithms are the ones that, for *any* initial point, produce a sequence of points $\{x^k\}$ that converges to a point of Ω.

The strongest condition that globally convergent algorithms can guarantee is the termination in a *finite* number of iterations. It is not common to find such algorithms, since they are limited to some peculiar class of problems, such as those with a convex quadratic objective function.

In the majority of cases, we have globally convergent algorithms that can only guarantee weaker conditions, such as the existence of a limit point of the sequence $\{x^k\}$ that belongs to the set Ω.
These algorithms are weaker than those that terminate in a finite number of steps, but they are still sufficient to guarantee that on a practical level they can produce a good estimate of a point of Ω after a sufficiently large number of iterations.

If an algorithm can only guarantee the termination in a finite number of steps or the existence of a limit point of the sequence $\{x^k\}$ that belongs to the set Ω *only* for starting points in a specific neighborhood of the optimal point x^*, we call it a *locally* convergent algorithm.

convergence rate

The other aspect normally analyzed in an algorithm is its *convergence rate*, a measure of the speed with which the produced sequence $\{x^k\}$ converges to a stationary point. Assuming the convergence of the sequence $\{x^k\} \subseteq \mathbb{R}^n$ to a point $x^* \in \Omega$, as in:

$$\lim_{k\to\infty} x^k = x^*,$$

the convergence rate of an algorithm is determined by the analysis of the asymptotic rate at which the error e_k defined as

$$e_k = ||x^k - x^*||$$

converges to zero. This error term measures the distance between the point obtained at the k-th iteration and the point x^*. It is worth underlying that authors sometimes characterize an algorithm convergence rate in terms of the error $e_k = \{f(x^k) - f(x^*)\}$, thus measuring the distance between the value assumed by the objective function at the k-th iteration and at the point x^*.

It is possible to find two main criteria in the literature for the analysis of the convergence rate of an algorithm:

- the R-rate, that is computed analysing the sequence of the roots of the error;
- the Q-rate, determined by the analysis of the sequence $\{\frac{e_{k+1}}{e_k}\}$.

The "R-" in the prefix stands for *root* and "Q-" stands for *quotient* because the terms are defined using the quotient between the errors of two successive iterations.

In the rest of the chapter we refer to the R-rate of convergence, for which we provide the following definitions:

Definition 1. *Given a sequence* $\{x^k\} \subseteq \mathbb{R}^n$ *converging to* $x^* \in \mathbb{R}^n$ *, we say that:*

- *if there exists* $c \in [0, 1)$ *such that, for sufficiently large k, we have:*

$$||x^{k+1} - x^*||^{1/k} \leq c,$$

  then $\{x^k\}$ *converges to* x^* *(at least) R-linearly;*

- $\{x^k\}$ *converges to* x^* *(at least) R-superlinearly if:*

$$\lim_{k\to\infty} ||x^{k+1} - x^*||^{1/k} = 0;$$

- *if there exists* $p > 1$ *and* $c \in [0, 1)$ *such that, for sufficiently large k, we have:*

$$||x^{k+1} - x^*||^{1/p^k} \leq c,$$

  then $\{x^k\}$ *converges to* x^* *R-superlinearly with R-rate (at least) p.*

2.1.2 Different types of converging algorithms

There are numerous ways to classify globally convergent algorithms that have the goal of finding stationary points. One of the most noticeable differences between algorithms is the information used to calculate s^k in the iteration

$$x^{k+1} = x^k + s^k.$$

Some algorithms only use the information contained in the objective function values, while others exploit the information contained in first and eventually second derivatives.

If the information about the first order derivative is available, we can further distinguish between the following classes of methods that use different strategies to update of point x^k at every iteration.:

- methods where the generic iteration can be described by

$$x^{k+1} = x^k + \alpha^k d^k$$

  where $d^k \in \mathbb{R}^n$ is called *search direction* and $\alpha^k \in \mathbb{R}$ is the *step-size*.

  These methods, known as *linesearch* algorithms, can generate the search direction d^k either by using only the information at the current point x^k or exploiting the information obtained at the previous iterations (i.e. by considering d^{k-1}, x^{k-1} and α^{k-1}). They are also known as *gradient-based* algorithms when information about the gradient is used.
  The step-size α^k is normally determined through a *linesearch* technique, a

procedure that gives the name to this class of algorithms, where the value of the objective function and the eventual information about the gradient is used to determine the magnitude of the movement along the direction d^k. The choice of both d^k and α^k is crucial in these algorithms and is designed in order to guarantee their global convergence.

- *trust region* methods, where the updates at each iteration are based on the creation of a model (often quadratic) that works as a surrogate of the objective function. This model is built in a region around the current point x^k, and the choice of s^k depends on the resolution of a constrained minimization problem where the objective function is given by such model. The mathematical models can be of different kinds: surrogate models, quadratic approximation models or trust-region models, and the precision with which they can locally approximate the function f is the first criterion used to evaluate their quality.

If the only information available is the value of the objective function, besides trust region and linesearch algorithms we find *direct methods*, which use only the comparison between function values in different points to determine candidate points, without attempting to develop a model of the function or an approximation of the gradient [140]. In particular, a local search to find a point where the objective function diminishes is implemented at any iteration [84, 102]. Among the various algorithms belonging to this family, the Nelder–Mead simplex algorithm [97] is probably the most renowned.

In the rest of this chapter we focus on some of the most popular gradient-based methods, namely algorithms for unconstrained minimization that exploit the information contained in the gradient of the objective function they try to minimize.

2.2 Gradient Descent

This section deals with the Gradient Descent method, which is one of the algorithms commonly used to solve unconstrained optimization problems. It has grown in popularity over the last few decades, becoming the most common way to optimize neural networks (especially in its *stochastic* version, specifically designed to deal with the abundance of data that characterize large-scale machine learning problems [22]).

2.2.1 The descent direction

The distinguishing feature of the Gradient Descent method is the computation of the direction d^k that is obtained considering an approximation of the objective function. The method deals with a *linear approximation* of $f(x^k + d)$, that is a function of the single variable d. Hence, under the hypothesis of f continuously differentiable, it is possible to write:

$$f(x^k + d) = f(x^k) + \nabla f(x^k)^T d + \beta_1(x^k, d),$$

where

$$\lim_{||d||\to 0} \frac{\beta_1(x^k, d)}{||d||} = 0.$$

Thus, the idea behind the Gradient Descent method is that of approximating the function $f(x^k + d)$ with the function $\psi_k(d)$ obtained as:

$$\psi_k(d) := f(x^k) + \nabla f(x^k)^T d$$

and to choose as the search direction d^k the one that minimizes $\psi_k(d)$ in the unit sphere. In other words, d^k is the solution of the following optimization problem:

$$\begin{aligned} min \quad & \psi_k(d), \\ \text{s.t.} \quad & ||d|| = 1, \end{aligned}$$

that is equivalent to

$$\begin{aligned} min \quad & \nabla f(x^k)^T d, \\ \text{s.t.} \quad & ||d|| = 1. \end{aligned} \tag{2.4}$$

Thanks to the *Schwarz inequality* we can write:

$$|\nabla f(x^k)^T d| \leq ||d||\, ||\nabla f(x^k),$$

where there is the equals sign if and only if

$$d = \lambda \nabla f(x^k)$$

with $\lambda \in \mathbb{R}$. The solution of problem (2.4) is therefore given by

$$d^k = -\frac{\nabla f(x^k)}{||\nabla f(x^k)||} \tag{2.5}$$

that defines the anti-gradient direction in x^k. This is the reason why Gradient Descent method is also known as *Steepest Descent*.

Taking into account (2.5), it is possible to describe Gradient Descent method with the following:

$$x^{k+1} = x^k - \tilde{\alpha}^k \frac{\nabla f(x^k)}{||\nabla f(x^k)||}, \tag{2.6}$$

that can be rewritten as:

$$x^{k+1} = x^k - \alpha^k \nabla f(x^k)$$

defining the step along the direction as $\alpha^k := \frac{\tilde{\alpha}^k}{||\nabla f(x^k)||}$.

It is worth highlighting that the local optimality of the direction $-\nabla f(x^k)$ depends on the choice of the norm, and that, given a point x^k every descending direction could be interpreted as the steepest descent direction for an appropriate norm [57].

Algorithm 2 Gradient Descent algorithm

1: Choose an initial point $x^0 \in \mathbb{R}^n$ and set $k = 0$
2: **while** $\nabla f(x^k) \neq 0$ **do**
3: Compute the search direction $d^k \in \mathbb{R}^n$ as $d^k = -\nabla f(x^k)$
4: Compute the step $\alpha^k \in \mathbb{R}$ along d^k
5: Set $x^{k+1} = x^k + \alpha^k d^k$
6: Set $k = k + 1$
7: **end while**

The scheme of the Gradient Descent algorithm is reported in Algorithm 2. Once the direction d^k is determined, we have the computation of the step α^k. Depending on the way α^k is determined, we distinguish between *inexact linesearch* and *exact linesearch*. In the former case, α^k is an *approximation* of a minimum of the function $\phi(\alpha) := f(x^k + \alpha d^k)$ which depends only on the scalar variable $\alpha \in \mathbb{R}^n$. In the latter case, α is determined as to find the *minimum* of $\phi(\alpha)$.

When dealing with an *inexact* linesearch, we can further distinguish between *monotone* methods, that ensure a reduction of the objective function f at every iteration, and *nonmonotone* methods, where occasional increase of the objective function are allowed.

As we show in the next section, to ensure the global convergence of Algorithm 2, the stepsize α^k needs to be chosen carefully.

2.2.2 Conditions on the stepsize for global convergence

The interest in the direction $-\nabla f(x^k)$ depends on the fact that, if the gradient is continuous, it represents a *continuous* descent direction with respect to x, and it assumes value zero if and only if x is a stationary point. This property is very important since it assures that, with an appropriate choice of the step α^k it is possible to prove the *global convergence* of the algorithm.

The following proposition shows a first set of conditions on the stepsizes α^k that guarantee the global convergence of the Gradient Descent algorithm:

Proposition 2.2.1. *Let $f : \mathbb{R}^n \to \mathbb{R}$ be a continuously differentiable function in $\mathbb{R}^n$ and let us assume that the gradient $\nabla f(x)$ is L-Lipschitz continuous for all $x \in \mathbb{R}^n$. Let $\{x^k\}$ be the sequence generated by Algorithm 2 where the stepsizes α^k satisfy:*

$$\sum_{k=0}^{\infty} \alpha^k = \infty, \qquad \sum_{k=0}^{\infty} (\alpha^k)^2 < \infty. \tag{2.7}$$

Then, either $\lim_{k\to\infty} f(x^k) = -\infty$ or every limit point $\bar{x}$ of $\{x^k\}$ is a stationary point of f.

One potential choice of the stepsize that satisfies conditions (2.7) is the following:

$$\alpha^k = \frac{1}{1+k} \tag{2.8}$$

where k indicates the iterations of Algorithm 2.

The noticeable benefit of this particular choice of the stepsize α^k is that it does not necessitate the evaluation of the objective function in any iteration. The drawback is that the stepsize goes to zero independently of the value of the objective function and might therefore go to zero too quickly, resulting in low rates of convergence.

From these considerations it follows that it makes sense to choose the stepsize according to (2.8) in all the instances where the computation of function evaluations is significantly expensive and that, in all other cases, it is preferable to choose a stepsize value that depends on the value of the objective function and that can yield to faster convergence rates.

Generally speaking, the stepsize α^k is required to produce a sufficent decrease of f, while producing at the same time a sufficient shift from the current point x^k (or α^k would go to zero too quickly).

Before showing the conditions on the the stepsize that guarantee the global convergence of the Gradient Descent algorithm, both in its monotone and nonmonotone version, we introduce the definition of *forcing function* and the definition of *level set*.

Definition 2. *A function $\sigma : \mathbb{R}^+ \to \mathbb{R}^+$ is a forcing function if for any sequence of numbers $t^k \subseteq \mathbb{R}^+$*

$$\lim_{k\to\infty} \sigma(t^k) = 0 \quad \textit{implies} \quad \lim_{k\to\infty} t^k = 0.$$

Definition 3. *With respect to the starting point x^0 of a minimization problem, we define the level set $\mathcal{L}_0$ in the following way:*

$$\mathcal{L}_0 = \{x \in \mathbb{R}^n : f(x) \leq f(x^0)\}$$

We now report three different sets of conditions on the values of α^k that guarantee the global convergence of Algorithm 2, both in its monotone and nonmonotone version. The proof can be found in [57].

Global convergence conditions for monotone methods

Proposition 2.2.2. *Let $f : \mathbb{R}^n \to \mathbb{R}$ be a continuously differentiable function in $\mathbb{R}^n$ and let us assume that the level set $\mathcal{L}_0$ is compact. Let $\{x^k\}$ be the sequence generated by Algorithm 2. We also assume that the following hold:*

(i) $f(x^{k+1}) \leq f(x^k)$ for all k;

(ii) if $\nabla f(x^k) \neq 0$ for all k, then

$$\lim_{k\to\infty} \frac{\nabla f(x^k)^T d^k}{\|d^k\|} = 0;$$

Then, either an index $\nu \geq 0$ such that $x^\nu \in \mathcal{L}_0$ and $\nabla f(x^\nu) = 0$ exists, or an infinite sequence is generated such that:

(a) $x^k \in \mathcal{L}_0$ for all k and $\{x^k\}$ admits limit points;

(b) every limit point of $\{x^k\}$ *belongs to* $\mathcal{L}_0$*;*

(c) sequence $\{f(x^k)\}$ *converges;*

(d) $\lim_{k\to\infty} \nabla f(x^k) = 0$;

(e) every limit point $\bar{x}$ *of* $\{x^k\}$ *satisfies* $\nabla f(\bar{x}) = 0$.

As mentioned before, once the direction d^k is set as the direction of the anti-gradient, the choice of the stepsize α^k becomes crucial for the satisfaction of the conditions described in Prop. 2.2.2.

In particular, since the direction $d^k = -\nabla f(x^k)$ is a descent direction, it must exists a stepsize $\bar{\alpha} > 0$ such that:

$$f(x^k + \alpha d^k) < f(x^k), \quad \forall \alpha \in (0, \bar{\alpha}].$$

A proper linesearch technique to find the stepsize α at each iteration is therefore crucial to satisfy condition (i), and is it also shown to play a role for the satisfaction of (ii).

In Prop. 2.2.2 it was assumed that the set $\mathcal{L}$ was compact, but there are situations where this assumption is false or cannot be known in advance. The proposition can be reformulated without including this assumption as follows.

Proposition 2.2.3. *Let* $f : \mathbb{R}^n \to \mathbb{R}$ *be a continuously differentiable function in* $\mathbb{R}^n$*. Let* $\{x^k\}$ *be the sequence generated by Algorithm 2. We also assume that the following holds:*

(i) $f(x^{k+1}) \leq f(x^k)$ *for all* k*;*

(ii) if $\nabla f(x^k) \neq 0$ *for all* k*, then*

$$\lim_{k\to\infty} \frac{\nabla f(x^k)^T d^k}{\|d^k\|} = 0;$$

Then, either an index $\nu \geq 0$ *such that* $x^\nu \in \mathcal{L}_0$ *and* $\nabla f(x^\nu) = 0$ *exists, or an infinite sequence* $\{x^k\}$ *with* $x^k \in \mathcal{L}_0$ *for all* k *is generated, such that one of the following is valid:*

(a) we have

$$\lim_{k\to\infty} f(x^k) = -\infty;$$

(b) the sequence $\{f(x^k)\}$ *is inferiorly limited and, in this case, we have that:*

(b.1) sequence $\{f(x^k)\}$ *converges;*

(b.2) $\lim_{k\to\infty} \nabla f(x^k) = 0$;

(b.3) every limit point $\bar{x} \in \mathcal{L}_0$ *(if any) satisfies*

$$\nabla f(\bar{x}) = 0.$$

One of the simplest ways to ensure the satisfaction of hypotheses i) and ii) of both Prop. 2.2.2 and 2.2.3 is to impose that the value α^k satisfies the Armijo condition:

$$f(x^k + \alpha^k d^k) \leq f(x^k) + \gamma \alpha^k \nabla f(x^k)^T d^k. \tag{2.9}$$

where $\gamma \in (0, 1)$ and d^k is a descent direction.
Such stepsize can be obtained with the *Armijo line search method*, one of the most popular line search algorithms that we report in the following scheme:

Algorithm 3 Armijo line search method

1: Given $\Delta^k > 0,\ \delta \in (0, 1),\ \gamma \in (0, 1/2)$
2: Set $\alpha = \Delta^k$
3: **while** $f(x^k + \alpha d^k) > f(x^k) + \gamma \alpha \nabla f(x^k)^T d^k$ **do**
4: Set $\alpha = \delta \alpha$
5: **end while**
6: Set $\alpha^k = \alpha$

Determining a stepsize α^k with the Armijo Line search is sufficient to satisfy the conditions (i) and (ii) in Prop. 2.2.2 and 2.2.3, thus guaranteeing the global convergence of Algorithm 2.

In both propositions it is assumed that the objective function monotonically decreases, and the lineasearch procedure described in Algorithm 3 is based on the concept of a "sufficiently reduction" of the objective function. There are several instances, though, where forcing monotonicity can have negative consequences on the behavior of the optimization algorithms, such as when an ideal value of the stepsize Δ^k is required to guarantee some properties about the convergence rate, but it does not yield to a monotone decrease of the objective function. In this scenario, the stepize obtained with the monotone line search of Algorithm 3 might be significantly smaller than the ideal Δ^k, producing smaller movements along the descent direction and ultimately yielding to a lower convergence rate.

In the following section report the conditions that guarantee the global convergence of the Gradient Descent algorithm without requiring the objective function to decrease at each iteration. In Sect. 2.4. we then provide an example of a nonmonotone, globally convergent, algorithm.

Global convergence conditions for nonmonotone methods

The following proposition holds.

Proposition 2.2.4. *Let $f : \mathbb{R}^n \to \mathbb{R}$ be a continuously differentiable function in $\mathbb{R}^n$ and let us assume that the level set $\mathcal{L}_0$ is compact. Let $\{x^k\}$ be the sequence generated by Algorithm 2. We also assume that the following hold:*

(i) $\lim_{k\to\infty} \left\| x^{k+1} - x^k \right\| = 0;$

(ii) there exists a subsequence $\{x^k\}_K$ *with* $0 \in K$ *such that:*

(ii-1) for every k *there exists* $j(k) \in K$ *such that*

$$0 < j(k) - k \leq M,$$

(ii-2) $\{f(x^k)\}_K$ *is non-increasing monotone, namely*

$$k_1, k_2 \in K \text{ and } k_2 > k_1 \text{ imply } f(x_{k_2}) \leq f(x_{k_1}),$$

(ii-3) if $\nabla f(x^k) \neq 0$ *for* $k \in K$*, then*

$$\lim_{k \in K, k \to \infty} \frac{\nabla f(x^k)^T d^k}{\|d^k\|} = 0,$$

Then, either an index $\nu \geq 0$ *such that* $x^\nu \in \mathcal{L}_0$ *and* $\nabla f(x^\nu) = 0$ *exists, or an infinite sequence is generated such that:*

(a) $\{x^k\}$ *admits limit points;*

(b) every limit point of $\{x^k\}$ *belongs to* $\mathcal{L}_0$*;*

(c) sequence $\{f(x^k)\}$ *converges;*

(d) $\lim_{k \to \infty} \nabla f(x^k) = 0$;

(e) every limit point $\bar{x}$ *of* $\{x^k\}$ *satisfies* $\nabla f(\bar{x}) = 0$.

To satisfy hypotheses i) and ii) of Prop. 2.2.4 it is sufficient to choose a value of α^k that satisfies the following condition:

$$f(x^k + \alpha^k d^k) \leq \max_{0 \leq j \leq min(k,M)} \{f(x^{k-j})\} + \gamma \alpha^k \nabla f(x^k)^T d^k. \tag{2.10}$$

where $\gamma \in (0, 1)$, d^k is a descent direction and M is an integer. The idea here is that it is not requested that the objective function decreases at *every* iteration, but it is sufficient that at each iteration it is smaller than the maximum value of the previous M iterations.
The linesearch procedure described in Algorithm 3 can be adapted to guarantee the satisfaction of (2.10) as shown in Algorithm 4.

Let us emphasize that the linesearch methods presented in this section are only two of several options that ensure the satisfaction of global convergence conditions. Interested readers can find several other algorithms for determining the appropriate stepsize in the literature [57, 99].

After focusing on the global convergence, in the next section we show how different choices for the stepsize α^k yield to different *rates* of convergence and, ultimately, to evolutions of the Gradient Descent method.

Algorithm 4 Nonmonotone Armijo line search method

1: Given $\Delta^k = a,\ \delta \in (0,1),\ \gamma \in (0,1), M > 1$
2: Set $\alpha = a$
3: **while** $f(x^k + \alpha d^k) > \max_{0 \le j \le min(k,M)} \{f(x^{k-j})\} + \gamma\alpha\nabla f(x^k)^T d^k$ **do**
4: Set $\alpha = \delta\alpha$
5: **end while**
6: Set $\alpha^k = \alpha$

2.3 On the convergence rate with different stepsizes

In this section we show how the choice of the step-size affects the convergence rate of the Gradient Descent algorithm. To simplify the discussion we will focus on the unconstrained minimization problem where the objective function is *strictly convex quadratic* as in:

$$\min_{x \in \mathbb{R}^n} f(x) = \frac{1}{2} x^T Q x - b^T x \tag{2.11}$$

where $Q \in \mathbb{R}^{n \times n}$ is a real symmetric positive definite matrix and $b \in \mathbb{R}^n$.

We show that stepsize that produces the largest decrease in the objective function does not necessarily yield to the best convergence rate, and that a stepsize computed considering second order derivatives can improve the efficiency of the method.

We then present the Barzilai-Borwein method, where the choice of the stepsize incorporates information of the second order derivatives without requiring their computation.

2.3.1 The "optimal" choice of the stepsize

In the ideal implementation of the Gradient Descent method, the direction d^k is computed as the anti-gradient direction, as shown in (2.5). Along this direction, the (seemingly) best stepsize α^k is the one obtained by solving the following unconstrained minimization problem:

$$\min_{\alpha} \phi(\alpha) = f(x^k + \alpha d^k),$$

or, in other words, by finding the value of α that produces the largest decrease of the value of the objective function along the descent direction d^k.

From an intuitive point of view, this ideal choice of d^k and α^k should be the one associated to the best performances of this method. However, this is not always the case. In fact, there are several examples in which this choice leads to sub-optimal convergence rate of the method and this represents one of the main downsides of the Gradient Descent method.

The following proposition, whose demonstration can be found in [57], shows that in the ideal scenario where the function to be minimized is *strictly convex quadratic* and the step α^k is computed minimizing exactly the function $\phi(\alpha) = f(x^k + \alpha d^k)$,

there are some initial points from which the Gradient Descent method produces a sequence of points $\{x^k\}$ whose rate of convergence is linear.

Proposition 2.3.1. *Consider the function f of problem* (2.11).
If α^k is obtained by exactly minimizing the function $f(x^k - \alpha d^k)$, i.e:

$$\alpha^k = -\frac{\nabla f(x^k)^T d^k}{(d^k)^T Q d^k} = \frac{||\nabla f(x^k)||^2}{\nabla f(x^k)^T Q \nabla f(x^k)},$$

the Gradient Descent method converges to the minimum of f, $x^* = 0$, and the following holds:

$$||x^{k+1} - x^*|| \leq \left(\frac{\lambda_M}{\lambda_m}\right)^{\frac{1}{2}} \left(\frac{\lambda_M - \lambda_m}{\lambda_M + \lambda_m}\right) ||x^k - x^*||$$

where λ_M and λ_m are the maximum and minimum eigenvalue of Q, respectively.

Prop. 2.3.1 shows that, when dealing with convex quadratic functions, Gradient Descent Method has a convergence rate that is at least linear and depends on the ratio λ_M/λ_m between the maximum and minimum eigenvalue of the Hessian matrix of $f(x)$. It is therefore possible to expect that the convergence rate of Gradient Descent method becomes worst as the difference between λ_M and λ_m increases, i.e. as the *ill-conditioning* of the matrix Q increases.

The following section shows an alternative way to compute the stepsize α^k that yields to better convergence properties of the method when the function to be minimized is quadratic.

2.3.2 A different choice criterion for the stepsize

The choice of α^k in the previous proposition was obtained by exactly minimizing the function $f(x^k - \alpha \nabla f(x^k))$ i.e. by choosing the step α^k in what seemed to be the best possible way. The following proposition shows the implications of a different choice of the step α^k .

Proposition 2.3.2. *Consider the function f of problem* (2.11). *The Gradient Descent method where the choice of the step is given by:*

$$\alpha^k = \frac{1}{\lambda_{k+1}} \tag{2.12}$$

where $\lambda_i, \quad i = 1, \ldots, n$ are the eigenvalues of the matrix Q, converges to the minimum of $f(x)$ x^* at most in n steps.

The proof of this proposition can be found in [57].

Differently from Prop. 2.3.1, with this choice of α^k, the convergence of the algorithm in a finite number of steps is guaranteed for *every* starting point. The difference between the results obtained with different choices of the step α^k yields to the following considerations:

- choosing α^k by trying to minimize the function $f(x^k - \alpha\nabla f(x^k))$ might not represent the best possible choice;
- different choices of the stepsize that contain information of the second derivative can improve the efficiency of the Gradient Descent method;
- since computing the eigenvalues can be extremely expensive, the particular choice of α^k in (2.12) is not feasible in most real-world applications.

In the following section, we describe the Barzilai-Borwein method, an improvement of the Gradient Descent method in which the choice of α^k is neither linked to the minimization along the search direction nor to the computation of the eigenvalues. This method manages to include the information of second order derivatives in the choice of α^k and has a better convergence rate than the Gradient Descend method.

2.3.3 The Barzilai-Borwein method: a new stepsize

The Barzilai-Borwein gradient method can be seen as a Gradient Descent method where the choice of the stepsize along the negative gradient direction is obtained by approximating the secant equation underlying Quasi-Newton methods.
The intuition was that the choice of the value of the stepsize α^k can be improved by exploiting the information of the second order, without having to compute the expensive second derivative of the function.

Newton's method for the resolution of unconstrained minimization problems can be described by the following iteration:

$$x^{k+1} = x^k - [\nabla^2 f(x^k)]^{-1}\nabla f(x^k)$$

Conversely, Quasi-Newton methods can be described by:

$$x^{k+1} = x^k - \alpha^k[B^k]^{-1}\nabla f(x^k)$$

where B^k is a suitable iteratively updated matrix that approximates (in some sense) the Hessian matrix.

Considering the same function of Prop. 2.3.1 and 2.3.2, namely:

$$f(x) = \frac{1}{2}x^T Qx - b^T x,$$

where $Q \in \mathcal{R}^{nxn}$ is a positive definite symmetric matrix and $b \in \mathcal{R}^n$. We have that:

$$\nabla f(x) = Qx - b$$

and, for any points x and y, we can write:

$$\nabla f(y) - \nabla f(x) = Q(y - x),$$

or, equivalently:

$$Q^{-1}[\nabla f(y) - \nabla f(x)] = y - x,$$

When the function is quadratic, it is therefore possible to describe the Quasi Newton method as the attempt to find a symmetric positive definite matrix such that the following equation holds:

$$B_k\, s = y$$

where

$$s = x^k - x^{k-1}, \quad y = \nabla f(x^k) - \nabla f(x^{k-1}).$$

This is equivalent to solve the following minimization problem:

$$\min_B \quad ||Bs - y||^2 \tag{2.13}$$

or, in other words, to find all the matrices B_k such that $||B_k s - y||^2 = 0$.

Until now, we have just described the general Quasi Newton method. The innovation proposed by Barzilai and Borwein in their paper in 1988 [13] was trying to solve problem (2.13) not for all the potential matrices B_k, but just for the ones that satisfy either $B_k = \alpha I$ or $B_k^{-1} = \frac{1}{\alpha} I$, in order not to deal with matrices.

Solving the problem for α allows to find a stepsize that is not linked to the value of the objective function, does not require the computation of eigenvalues and still manages to contain information that derive from the Hessian matrix.
In particular, the two stepsizes identified by the authors, and that can be found in the scheme of the Barzilai-Borwein method adescribed in Algorithm 5, are

$$\alpha^k = \alpha^1 = \frac{s^T y}{s^T s},$$

obtained by minimizing $||B_k\, s - y||$ when $B_k = \alpha I$ and

$$\alpha^k = \alpha^2 = \frac{y^T y}{s^T y},$$

obtained by setting $(B_k)^{-1} = \frac{1}{\alpha} I$ and minimizing the quantity $||s - \frac{1}{\alpha} y||$ [56].

When proposing it in 1988, Barzilai and Borwein showed that the BB method is faster than the classic Gradient descent method on an example with a quadratic function, and they proved the R-superlinear convergence rate of the BB method when the function is quadratic and the number of dimensions is equal to two [13]. Subsequent works proved the global convergence of the BB method in the case of strictly convex quadratic functions with any number of variables [111] and provided the following proposition, whose proof can be found in [37]:

Proposition 2.3.3. *Let f be a strictly convex quadratic function. Let $\{x^k\}$ be the sequence generated by Algorithm 5. Then, either $\nabla f(x^k) = 0$ for some finite k, or the sequence $\{||\nabla f(x^k)||_2\}$ converges to zero R-linearly.*

Algorithm 5 Barzilai-Borwein Scheme

1: Choose an initial point $x^0 \in \mathbb{R}^n$, an initial stepsize α^0 and set $d^0 = -\nabla f(x^0)$.
2: Produce a new point
$$x^1 = x^0 + \alpha^0 d^0,$$
and set $k = 1$.
3: If x^k is a stationary point, stop.
4: Select $d^k = -\nabla f(x^k)$.
5: Select the scalar α^k either by
$$\alpha^k = \alpha^1 = \frac{s^T y}{s^T s}$$
or by
$$\alpha^k = \alpha^2 = \frac{y^T y}{s^T y}$$
with
$$s = x^k - x^{k-1}, \quad y = \nabla f(x^k) - \nabla f(x^{k-1}).$$
6: Produce a new point
$$x^{k+1} = x^k + \alpha^k d^k,$$
set $k = k + 1$ and go back to Step 3.

In the general case of non-quadratic functions, however, it is not possible to prove the convergence of the BB. In fact, α^1 and α^2 could assume both extremely big and extremely small values, thus requiring the stepsize α^k computed at Step 4 to be adjusted in order to satisfy the following condition:

$$0 < \alpha^\ell \leq \alpha^k \leq \alpha^u \quad \text{for all } k$$

where α^ℓ and α^u are predetermined numbers.

In the next section, we show how the BB method can be modified to guarantee the global convergence in the general case of non-quadratic functions.

2.4 A globally convergent versione of the BB method

In this section we present the globalization strategy for the Barzilai-Borwein method proposed in 2002 by Grippo and Sciandrone in [56]. We then describe the NonMonotone Stabilization (NMS) algorithm that embeds the nonmonotone globalization strategy and is proven to be globally convergent.

The idea behind this algorithm is that of trying to combine two things: the preservation of the satisfaction of the conditions that guarantee the global convergence of local optimization algorithms and the willingness to perform as many iterations as possible with the BB stepsize. As we discussed in Sect. 2.3.3, indeed, implementing

the Gradient Descent method where the value of the stepsize α^k is linked to the secant equation underlying Quasi-Newton methods, makes it extremely efficient in the strictly convex quadratic case since it guarantees the convergence of the algorithm in n steps. It also appears that allowing a sequence of steps from the unmodified method improves the behavior of globalization approaches for the BB method in the *nonconvex* case.
At the same time, using such stepsize does not yield to the decrease of the objective function at each iteration and does not guarantee the convergence of the BB algorithm in the nonconvex case. Hence, when necesary, a procedure to change the step length is required.

The globalization strategy proposed in the NMS algorithm does exactly this. The algorithm tries to perform several iterations of the Gradient Descent method where the stepsize α^k is the one of the Barzilai-Borwein method, unless it is too small or too big - and in this case it is modified so to assume a value included in a pre-determined acceptable range. Since this choice of the stepsize does not yield to a constant decrease of the objective function, it is impossible to know the quality of the points produced by this choice without going through several iterations. The number of such iterations is denoted by N, which represents the patience (or budget) granted to the BB method and is determined at the beginning of the algorithm.

We now briefly describe the nonmonotone stabilization strategy of the NMS algorithm, before showing its scheme in Algorithm 7. In the article proposed by Grippo and Sciandrone, the authors present two different versions of this algorithm: NMS1 and NMS2, conceptually very similar. We describe both of them and report only the scheme of NMS2 for brevity.

2.4.1 The nonmonotone stabilization strategy

Indicating by x^k the point considered at the major iteration, the algorithm NMS1 computes a descent direction d^k satisfying a suitable condition and performs N steps using the BB stepsize, reaching a tentative point z_N^k. If this point turns out to be acceptable, it is set as the point x^{k+1} and the algorithm continues. Otherwise the new point is rejected, the algorithm goes back to x^k and computes the stepsize λ^k along the descent direction using a nonmonotone linesearch technique. The NMS2 version of this algorithms consists in performing the check of whether the point is acceptable or not for each of the N tentative point obtained using the BB stepsize, not limiting this check to the last of these points.

Thus, the rationale behind this algorithm is trying to take advantage of the BB stepsize which does not require the resolution of any minimization problem or the use of a linesearch technique, and is not time consuming. If, after some iterations, this *easy* way of finding new points has not yielded to an *acceptable point*, the algorithm returns to the starting point of the beginning of the iteration and uses a linesearch method to determine the stepsize.

It is now worth specifying what the expression *acceptable point* means. A point is considered to be acceptable if it satisfies some conditions that are the ones normally requested to guarantee the global convergence of local optimization algorithms. In the proposed algorithm, the points are requested to asymptotically get close to each other. In other words, it is requested that the new points produced at each iteration become closer and closer to each other as the number of iteration grows, as shown in Sect. 2.2.2. At the same time, the conditions requested ensure the convergence of the method to a stationary point.

Denoting with k the current iteration, z_i the tentative point, σ the forcing function and with N and M two integers, the condition that needs to be satisfied to accept the tentative point is the following:

$$\textbf{if} \quad f(z_i) \leq \max_{0 \leq j \leq min(k,M)} \{f(x^{k-j})\} - \max_{1 \leq h \leq i} \{\sigma(||z_h^k - x^k||)\} \textbf{ then}$$

you can accept the tentative point.

We can give the following interpretation to this condition: if the tentative points obtained using the BB stepsize are close from the starting point x^k, a small decrease of the objective function can be considered sufficient in order to accept the tentative point considered. If, on the other end, using the BB stepsize the algorithm has moved to points that are far from the one at the beginning of the iteration, the tentative point can be accepted only if it guarantees a significant decrease of the objective function. In particular, the condition considers the points that is *furthest* from x^k, hence the *max* in the second term on the right side of the condition. Since the method is nonmonotone, the objective function of the tentative point is not compared to the one of the point at the beginning of the iteration, but to the maximum objective function of the last M iteration, thus allowing for small increases of the objective function between different iterations.

We can think of the NMS algorithm as if made up of two parts. The first part is the one where, starting from a point x^k, several tentative points - whose number depends on the budget allocated to this exploration phase - are obtained using the BB stepsize, and for each of them the acceptance test is performed. If none of the points satisfies the acceptance test, the algorithm moves to its second phase. This second, standard in way, phase consists in determining a descent direction d^k that satisfies some suitable condition and in performing a nonmonotone linesearch. The conditions imposed on the choice of the descent direction and the acceptance criteria of the stepsize inside the linesearch are the elements that guarantee the convergence of the algorithm.

The description and scheme of this “standard” part of the algorithm are provided in the following section.

2.4.2 The nonmonotone linesearch

The choice of direction d^k used in the standard algorithm is linked to the satisfaction of the following condition:

Condition 1. *There exists positive numbers c_1, c_2 such that, for all k it is true that:*

(i) $||d^k|| \leq c_1||\nabla f(x^k)||$;

(ii) $\nabla f(x^k)^T d^k \leq -c_2||\nabla f(x^k)||^2$.

It is easy to see that any direction d^k that satisfies Condition 1 is a descent direction.

Starting from the point x^k, once the descent direction d^k is determined, the nonmonotone linesearch take place. The sequence produced by the linesearch needs to satisfy the following condition:

Condition 2. *Let $\{x^k\}$ be a sequence of points and let $\{d^k\}$ be a sequence of search directions. Assume that $K \subseteq \{0, 1, \dots\}$ is a subset such that $x^{k+1} = x^k + \lambda^k d^k$ for all $k \in K$, where $\lambda^k \in \mathbb{R}$ is computed through the linesearch procedure. Then:*

(i) for every $k \in K$ we have

$$f(x^k + \lambda^k d^k) \leq \max_{0 \leq j \leq min(k,M)} \{f(x^{k-j})\} - \sigma_l(\lambda^k ||d^k||),$$

where $M \geq 0$ is a prefixed integer, and $\sigma_l : \mathbb{R}^+ \to \mathbb{R}^+$ is a forcing function;

(ii) if K is an infinite subset, if the sequence $\{f(x^k)\}$ converges and the subsequence$\{x^k\}_K$ is bounded, it follows that

$$\lim_{k \to \infty, k \in K} \frac{\nabla f(x^k)^T d^k}{||d^k||} = 0.$$

We can see that Condition 2(i) is the one that guarantees a sufficient decrease of f and at the same time ensures that the stepsize is bounded so that, in the case of convergence, the following holds: $\lambda^k ||d^k|| \to 0$. Condition 2(ii) expresses the convergence properties of the linesearch procedure, and implies that the procedure produces "sufficiently large" stepsizes.

Algorithm 6 describes the nonmonotone linesearch procedure.

In this linesearch, the first attempted stepsize is $\lambda = 1$. What happens next depends on the satisfaction of the condition on the sufficient decrease of the objective function. If the new point obtained by moving along the descent direction with the initial stepsize does not yield to a sufficient decrease of the objective function (*Step 2*), the stepsize is decreased until the condition is satisfied. If, on the other end, for $\lambda = 1$ there is already a sufficient decrease of f with respect to a reference value, there is another crossroad: if the norm of the direction is too big, or if the value of the new objective function is greater than the one of the point x^k, the algorithm

Algorithm 6 Nonmonotone linesearch algorithm

Data: γ_1 and γ_2 that satisfy:

$$0 \leq \gamma_1 < 1, \quad 0 \leq \gamma_2, \quad 0 < \gamma_1 + \gamma_2$$

and

$$F^k = \max_{0 \leq j \leq \min(k,M)} [f(x^{k-j})], \quad \Delta^k = \Psi\big(\frac{|\nabla f(x^k)^T d^k|}{||d^k||}\big)$$

where $M > 0$ is an integer and $\Psi : \mathbb{R}^+ \to \mathbb{R}^+$ is a forcing function

Step 0. **if** $\gamma_2 = 0$ **then**
Choose a number $\bar{\lambda} \geq 1$
else
Set $\bar{\lambda} = +\infty$
end if

Step 1. Set $\lambda = 1$

Step 2. **while** $f(x^k) + \lambda d^k > F^k + \gamma_1 \lambda \nabla f(x^k)^T d^k - \gamma_2 \lambda^2 ||d^k||^2$ **do**
Choose $0 \leq \theta \leq 1$
Set $\lambda = \theta \lambda$
end while

Step 3. **if** $\lambda < 1$ **then**
Set $\lambda^k = \lambda$ and exit
end if

Step 4. **if** $||d^k|| \geq \Delta^k$ or $f(x^k + d^k) \geq f(x^k)$ **then**
Set $\lambda^k = 1$ and exit
else
Choose $\sigma > 1$
end if

Step 5. **while** $\sigma\lambda \leq \bar{\lambda}$ **and**

$$f(x^k + \sigma\lambda d^k) < \min\{f(x^k + \lambda d^k), f(x^k) + \gamma_1 \sigma\lambda \nabla f(x^k)^T d^k - \gamma_2 (\sigma\lambda)^2 ||d^k||^2\}$$ **do**

Set $\lambda = \sigma\lambda$
Choose $\sigma > 1$
end while

Step 6. Set $\lambda^k = \lambda$ and exit

accepts the stepsize $\lambda = 1$ and exits (*Step 4).* If none of this happens, the direction d^k is seen as a "promising" direction, and the algorithm tries to achieve a bigger reduction of the objective function by iteratively increasing the value of the stepsize until the condition is no longer satisfied (*Step 5*).

In [56] the authors demonstrate that this algorithm satisfies Condition 2 described above, thus guaranteeing its convergence properties.

In the following section we describe how the NMS algorithm combines the principle of trying to exploit the BB stepsize whenever possible with the linesearch algorithm that has just been described.

2.4.3 NonMonotone Stabilization Algorithm 2 (NMS2)

The scheme of the NMS2 algorithm is presented in Algorithm 7.

Algorithm 7 NMS2 algorithm

Data: $x^0 \in \mathbb{R}^n$, integers $N \geq 1, M \geq 0, k = 0$ and a forcing function $\sigma : \mathbb{R}^+ \to \mathbb{R}^+$
while $\nabla f(x^k) \neq 0$ **do**
Step 1. Compute a direction d^k satisfying Condition 1
Set *linesearch* = **true**
Step 2(a). Set $z_0^k = x^k$
Step 2(b). **for** $i = 1, \ldots, N$ **do**
$z_i^k = z_{i-1}^k + p_i^k$ where p_i^k is a suitable search direction
if $f(z_i^k) \leq \max_{0 \leq j \leq min(k,M)} \{f(x^{k-j})\} - \max_{1 \leq h \leq i} \{\sigma(||z_h^k - x^k||)\}$ **then**
Set $x^{k+1} = z_i^k$
Set *linesearch* = **false** and exit from Step 2.
end if
end for
Step 3. **if** *linesearch* = **true then**
Compute the stepsize λ^k along d^k by means of a linesearch algorithm ensuring Condition 2
Set $x^{k+1} = x^k + \lambda^k d^k$
end if
Step 4. Set $k = k + 1$
end while

The algorithm starts with an initial point x^0, a given forcing function σ and with the integers N and M that represent the "budget" allocated to the exploration of the BB stepsize and the "memory term" that appears in Condition 2, respectively.

The algorithm stops as soon as the obtained point x^k is a stationary point. To generate new points the algorithm does the following: firstly, it generates a descent direction d^k that satisfies Condition 1. It then produces a finite number of tentative points using the BB method. In particular, p_1^k is defined as the descent direction d^k

defined at the previous step (i.e. $d^k = p_1^k$) and the scaled steepest descent directions derived using the BB equations make up the rest of the p_i^k.

For each of the tentative points the satisfaction of the criterion at *Step 2(b)* is tested. This criteria assesses the objective function's decrease in comparison to a reference value. As soon as one of the points satisfies the test, the algorithm sets x^{k+1} as the tentative point z_i^k that can be accepted. If the test fails for all the N points obtained with the BB method, the algorithm backtracks to x^k and then proceeds with the generation of the new point x^{k+1} by employing the nonmonotone linesearch that we described in the previous section (*Step 4*). In the NMS1 version of the NonMonotone Stabilization Algorithm that has the exact same convergence properties of NMS2, only the last of the N points obtained using the BB stepsize (*Step 2* of the algorithm) is considered as a potential, tentative point, and is the only one for which the acceptance test is performed.

Both NMS1 and NMS2 are globally convergent algorithms. The following two proposition, whose proof can be found in [56], show the converging properties of the produced sequence of points. Prop. 2.4.1 indicates the converging properties of the sequence x^k, whereas Prop. 2.4.2 establishes the converging properties of the sequence of the tentative points. The latter refers specifically to algorithm NMS2, but it can easily be extended to algorithm NMS1 by always setting $j = N$.

Proposition 2.4.1. *Let $\{x^k\}$ be the sequence generated by either Algorithm NMS1 or algorithm NMS2 and suppose that the algorithm does not terminate. Then, every limit point of $\{x^k\}$ is a stationary point of f, which is not a maximum point.*

Proposition 2.4.2. *Let $\{x^k\}$ be the sequence generated by algorithm NMS2 and suppose that the algorithm does not terminate. Let z_i^k, for $i = 1, \ldots, N$ be the points generated at Step 2(b) when the test at Step 2(b) is satisfied for some $1 \leq j \leq N$ so that $x^{k+1} = z_j^k$. Then, every limit point of each sequence $\{z_i^k\}$ is a limit point of $\{x^k\}$ and hence a stationary point of f, which is not a local maximizer.*

Chapter 3

Optimization with noisy function

In this Chapter we consider the optimization problem introduced in Chapter 2:

$$\min_{x \in \mathbb{R}^n} f(x) \tag{2.1}$$

in the situation where function evaluations are affected by noise. There are several sources of noise that affect many real-world optimization problems, like the use of stochastic simulation models in place of the objective function [42], or the use of stochastic differential equations to model phenomena in various fields, ranging from financial risk [88] to the electrophoretic separation of DNA molecules [27]. Sometimes noise is generated because of physical measurement limitations or when function evaluations involve the approximate solution of a numerical problem [6]. Deterministic simulation models could also be noisy because of the occurrence of various numerical errors.
We model the noisy scenario considering the presence of an additive noise affecting the sampled function values $f(x)$. We have that:

$$f(x) = \phi(x) + \epsilon(x) \tag{3.1}$$

where $f(x)$ indicates the perceived objective function value, $\phi(x)$ represents the *real* objective function value and the error term $\{\epsilon(x)\}$ denotes a discrete random field modeling the additive noise on the *sampled* function values with the following properties: $\epsilon(x) \sim N(0, \lambda^2)$ and $E[\epsilon(x_i)\,\epsilon(x_j)] = 0$ for $x_i \neq x_j$. [1]

In absence of noise we have that $f(x) = \phi(x)$ and we will often refer to the *real* objective function using the common notation $f(x)$ instead of the less used $\phi(x)$.

Fig. 3.1 shows the example of a function affected by noise of varying magnitude in each subplot. The standard deviation λ of the noise is reported in the legend of each plot.

Solving optimization problems in the presence of noise is not an easy task and is a popular research topic. The impact of noise in convex optimization has been

[1] The noise is modeled on the sampled function values for an easier characterization in the discrete domain of uncorrelated noise values.

studied in [96], while algorithms to estimate the numerical noise can be found in [91, 92]. Finally, adaptations of Taylor based methods when function evaluations are affected by noise are provided in [15].

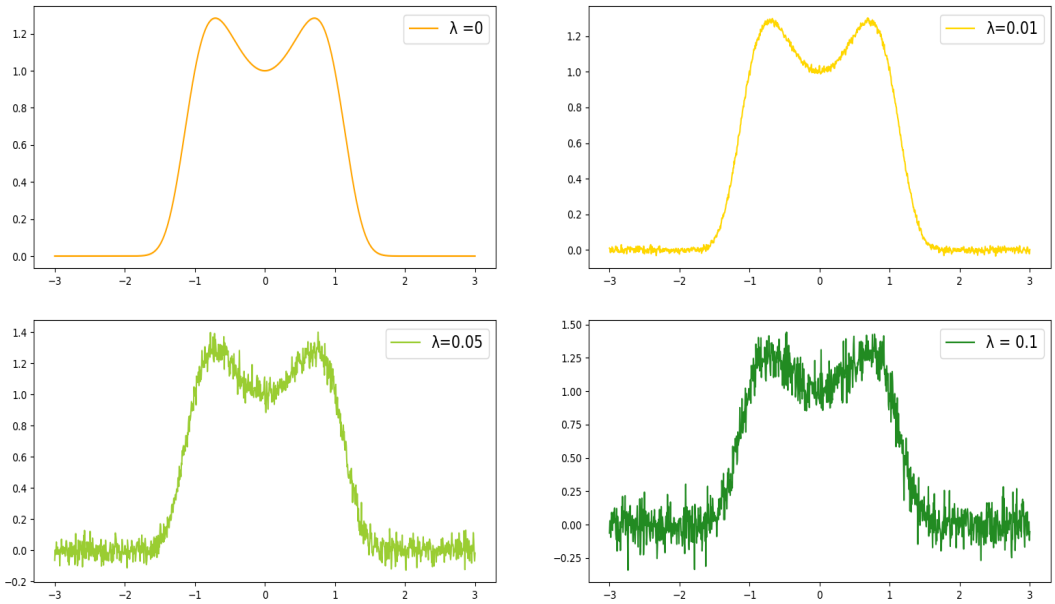

Figure 3.1. Example of the function $f(x) = e^{x^2(1-x^2)}$ affected by noise.

In this chapter - and in the rest of this thesis - we distinguish between two different case studies regarding function evaluations affected by noise that require two different approaches:

- situations where objective function evaluations can be obtained in a cheap and fast way;
- instances in which evaluating the objective function is very expensive in terms of time and resources, such as when it requires the use of an external simulation software (e.g. in simulation-based optimization problems).

In the former scenario, we can try to use objective function evaluations to somehow filter the noise and build an estimate of the gradient 2.2. This enables the use of gradient-based methods for solving problem 1.1 despite not having the possibility of computing the real value of $\nabla f(x)$.

When the the evaluation of the objective function is significantly expensive, this approach would not be feasible. In this case, the effort will be in the creation of a model that approximates the objective function and gradient-based methods will be used considering the gradient of such model. In particular, we will focus on the employment of neural networks for the creation of the approximating model.

The remainder of the chapter is organized as follows: Sect. 3.1 focuses on the review of well-known methods from the literature for building gradient approxima-

tions, whereas in Sect. 3.2 we introduce the basic theory behind neural networks and their role as universal approximators.

3.1 Gradient Approximation methods: an overview

In this section we focus on several methods proposed in the literature for constructing a good approximation of the gradient 2.2 using only information from the objective function. All of these methods use function values of points that are in the neighborhood of x and differ in three ways: the number of points used, how these points are chosen and how they are used to build the gradient approximation.

In the following sections a description of various, widely adopted, methods is provided, along with some of their theoretical properties.

We denote with $g(x)$ any estimate of $\nabla f(x)$ and with $g_i(x)$ its i_{th} component. We choose not to use a different subscript for every approximation since we believe it would make the notation too heavy. We also believe that it will be easy for the reader to understand what scheme $g(x)$ is referring to.

3.1.1 The Finite Difference scheme

One of the approaches to the computation of a gradient estimate derives from Taylor's theorem and is known as finite differencing [99]. The idea behind this method is intuitive: to estimate the derivative of a function in a point x you can compute small and *finite* perturbation of the values of x and then observe the *differences* in the function values. The approximation of the derivative, that is known to measure of the sensitivity to change of a function determined by extremely small changes of the independent variable, is then obtained by computing the ratio between such difference and the magnitude of the perturbation of x.

The finite differences method exists in two forms: the *forward finite difference* approximation, or *FFD*, and the *central finite difference* approximation - *CFD*, both of which are described in this section.

Forward Finite Difference scheme

Indicating with $\{e_i\}$ the canonical base of $\mathbb{R}^n$, we can write the approximation of the gradient obtained with FFD in the following way:

$$g_i(x) = \frac{f(x + \sigma e_i) - f(x)}{\sigma}, \quad i = 1, \ldots, n. \tag{3.2}$$

It has been stated that this estimate derives from Taylor's theorem. We now justify this claim and find a bound of the distance between $g(x)$ and $\nabla f(x)$.

Proposition 3.1.1. *If the function $f : \mathbb{R}^n \mapsto \mathbb{R}$ is continuously differentiable and its Gradient is L-Lipschitz continuous for all $x \in \mathbb{R}^n$ we have that:*

$$||g(x) - \nabla f(x)|| \leq \sqrt{n}\frac{L\sigma}{2}. \tag{3.3}$$

where $g(x) = \sum_{i=1}^{n} g_i(x)\, e_i$.

Proof. Under the assumptions of Prop. 3.1.1, Taylor's theorem with remainder in the integral form in the case of k=1 states that:

$$f(x) = f(a) + (x-a)f'(a) + \int_a^x (x-t)f''(t)dt$$

We can therefore write:

$$f(x + \sigma e_i) = f(x) + \sigma\, \nabla f(x)^T e_i + \sigma^2 \int_0^1 (1-t) e_i^T \nabla^2 f(x + t\sigma e_i) e_i\, dt$$

The distance between $g_i(x)$ computed as 3.2 and $\frac{\partial f}{\partial x_i}(x)$ can be written as

$$\begin{aligned}
&\left| g_i(x) - \nabla f(x)^T e_i \right| = \left| \frac{f(x + \sigma e_i\ ) - f(x)}{\sigma} - \nabla f(x)^T e_i \right| \\
&= \left| \sigma \int_0^1 e_i^T \nabla^2 f(x + t\sigma e_i) e_i\, (1-t)\, dt \right| \leq \frac{1}{\sigma} \int_0^1 \left| e_i^T \nabla^2 f(x + t\sigma e_i) e_i \right| (1-t) dt \\
&\leq \frac{1}{\sigma} L \left(-\frac{1}{2}\,(1-t)^2 \Big|_0^1 \right) = \frac{L\sigma}{2}.
\end{aligned} \tag{3.4}$$

In $\mathbb{R}^n$, remembering that $||x|| = (\sum_{i=1}^n x_i^2)^{\frac{1}{2}}$ we have:

$$||g(x) - \nabla f(x)||^2 = \sum_{i=1}^{n} \left(g_i(x) - \nabla f(x)^T e_i \right)^2 \leq \sum_{i=1}^{n} \left(\frac{L\sigma}{2} \right)^2 = n \left(\frac{L\sigma}{2} \right)^2. \tag{3.5}$$

From which derives 3.3:

$$||g(x) - \nabla f(x)|| \leq \sqrt{n}\frac{L\sigma}{2}.$$

□

When looking at (3.3), the reader may be tempted to choose a step-size σ as small as possible, but should keep in mind the numerical limitations that are faced if this algorithm is to be implemented on a real computer.

In fact, obtaining the gradient estimate with the finite difference formula requires the computer to calculate sums and differences of floating-point numbers and then dividing the result by a small number, σ. This is the same as multiplying the result by a huge number, with the consequence that any rounding errors in the numerator will be magnified by the multiplication. Since this amplification increases as σ

decreases, its value should be chosen to minimize the total error, which includes both the truncation error (which is decreasing in σ) and the round-off error that is generated when computing arithmetic operations with floating-point integers (and is increasing in σ). In [99], Chapter 7, the author proposes a reasonable way to choose the value of σ based on this trade-off, but it is worth noticing that in practice the value of the Lipschitz constant L that appears in the truncation error is rarely known. When it is not, the user is left to the choice of either trying to estimate the value of the Lipschitz constant, or to manually adjust the value of σ based on the results of his experiments.

As if this were not enough, the choice of the step-size σ becomes even more critical when function evaluations are affected by noise, as we discuss in the following section.

FFD in the presence of noise

If function evaluations are affected by noise as modelled in (3.1), we can write the approximation of the gradient $g(x)$ obtained with FFD in the following way:

$$g(x) = \Big[g_1(x), g_2(x) \ \dots \ g_n(x)\Big]^T$$

where:

$$g_i(x) = \frac{\phi(x + \sigma e_i) + \epsilon(x + \sigma e_i) - \phi(x) - \epsilon(x)}{\sigma}. \tag{3.6}$$

Let us now analyze the estimation error in presence of noise in terms of accuracy (mean value) and precision (variance). The accuracy evaluates the estimate bias, i.e. the systematic source of the error, like the limited number of function realizations available to build the estimate, whereas the precision is the dispersion of the estimation error around its mean value and evaluates the variability of the statistic source of the error.
Let

$$e_{\text{FFD}}(x) = g(x) - \nabla\phi(x)$$

be the estimation error and let $\{e_i\}$ be the canonical base of $\mathbb{R}^n$. We then have that:

$$E[e_{\text{FFD}}(x)] = \sum_{i=1}^{n} \frac{\phi(x + \sigma e_i) - \phi(x)}{\sigma} e_i - \nabla\phi(x) \tag{3.7}$$

since the expected value of every noisy term $\epsilon(\cdot)$ is zero, and

$$var[e_{\text{FFD}}(x)] = n \frac{(2\,\lambda)^2}{\sigma^2} \tag{3.8}$$

where $var[z]$, $z \in \mathbb{R}^n$ with $E[z] = 0$, indicates the trace of the covariance matrix $E[z\,z^T]$.

Now, for functions f as in Prop. (3.1.1),we have that

$$\|E[e_{\text{FFD}}(x)]\| \leq \sqrt{n}\,\frac{L\,\sigma}{2}.$$

Therefore, as the step-size $\sigma \to 0$ the expected value of error goes to zero as well *on average*, but its variance increases without bound as $\mathcal{O}\left(1/(\sigma)^2\right)$. This trade-off between precision and accuracy shows with even more clarity that the choice of the value of the step-size is anything but simple, especially in the presence of noise.

While in the noise-free setting it was possible to put a deterministic bound on the estimation error $e_{\text{FFD}}(x)$, in the presence of noise we can only provide a statistical analysis of it. The following proposition holds:

Proposition 3.1.2. *If the function $f : \mathbb{R}^n \mapsto \mathbb{R}$ is twice continuously differentiable and its Gradient is L-Lipschitz continuous for all $x \in \mathbb{R}^n$, if function evaluations are affected by noise as modelled in (3.1) we have that, with probability α:*

$$||g(x) - \nabla\phi(x)|| \leq \sqrt{n}\frac{L\sigma}{2} + \mathcal{P}(\alpha)\frac{\sqrt{n}\,(2\,\lambda)}{\sigma}. \tag{3.9}$$

where $g(x) = \sum_{i=1}^{n} g_i(x)\,e_i$ and $\mathcal{P}(\alpha)$ is the α-quantile of the standard gaussian distribution.

The proof can be derived directly from the one of Prop. 3.1.1 and basic statistical theory and is not reported here for brevity.

In other words, Prop. 3.1.2 states that we can bound the norm of the difference between the real gradient and the estimate with the sum of the expected value of the error and its standard deviation multiplied by a factor that depends on the desired confidence.
For example, we could say that with probability 95%

$$||g(x) - \nabla\phi(x)|| \leq \sqrt{n}\frac{L\sigma}{2} + 1.645\frac{\sqrt{n}\,(2\,\lambda)}{\sigma}.$$

From this formula we can see that, if the information on the Lipschitz constant of the gradient L and the value of the standard deviation λ were available, one could choose the right value of σ that allows to control the bound on $||g(x) - \nabla\phi(x)||$.

In the next two sections we will extend the analysis performed for the FFD scheme to the Central Finite Difference Scheme, which is more precise and more expensive in terms of function evaluations, and we will discuss its features both in the noise-free and in the noisy settings.

Central Finite Difference scheme

A more accurate way of computing the gradient approximation using the Finite Difference scheme is the CFD formula.
Denoting with $g_i(x)$ the i_{th} component of the estimate, the CFD approximation is:

$$g_i(x) = \frac{f(x + \sigma e_i\ ) - f(x - \sigma e_i)}{2\,\sigma}, \quad i = 1, \ldots, n. \tag{3.10}$$

The number of function evaluations used by this scheme is $2n$, almost twice as big as the one of the FFD scheme that uses a budget of $n + 1$ function evaluations. On the other hand, it guarantees a higher accuracy, as we show in this section. The trade-off between the number of function evaluations and the precision of the estimate is something that exists for all the gradient estimation schemes that can use a varying number of function evaluations.

Let us now derive a bound on the distance between this estimate and the real value of the gradient, under the assumption of f twice continuously differentiable and the H-Lipschitz continuity of its Hessian for all $x \in \mathbb{R}^n$.

Proposition 3.1.3. *If the function $f : \mathbb{R}^n \mapsto \mathbb{R}$ is twice continuously differentiable and its Hessian is H-Lipschitz continuous for all $x \in \mathbb{R}^n$ we have that:*

$$||g(x) - \nabla f(x)|| \leq \sqrt{n}\frac{M\sigma^2}{6}. \tag{3.11}$$

where $g(x) = \sum_{i=1}^{n} g_i(x)\, e_i$.

Proof. Under the assumptions of Prop. 3.1.3, Taylor's theorem with remainder in the integral form in the case of k=2 states that:

$$f(x) = f(a) + (x-a)f'(a) + (x-a)^2\,\frac{1}{2}f''(a) + \frac{1}{2}\int_a^x (x-t)^2 f'''(t)dt$$

We can therefore write:

$$f(x+\sigma e_i) = f(x)+\sigma\,\nabla f(x)^T e_i + \frac{1}{2}\sigma\, e_i^T \nabla^2 f(t) e_i\,\sigma + \frac{1}{2}\int_0^1 \frac{\partial^3 f(x+\sigma t e_i)}{\partial x_i^3}\sigma^3(1-t)^2 dt$$

and

$$f(x-\sigma e_i) = f(x)-\sigma\,\nabla f(x)^T e_i + \frac{1}{2}\sigma\, e_i^T \nabla^2 f(t) e_i\,\sigma - \frac{1}{2}\int_0^1 \frac{\partial^3 f(x-\sigma t e_i)}{\partial x_i^3}\sigma^3(t-1)^2 dt.$$

The distance between $g_i(x)$ computed as 3.10 and $\frac{\partial f}{\partial x_i}(x)$ can be written as

$$\begin{aligned}
&\left|g_i(x) - \nabla f(x)^T e_i\right| = \left|\frac{f(x+\sigma e_i\ ) - f(x-\sigma e_i)}{2\,\sigma} - \nabla f(x)^T e_i\right| \\
&= \left|\frac{1}{4\sigma}\int_0^1 \frac{\partial^3 f(x+\sigma t e_i)}{\partial x_i^3}\sigma^3(1-t)^2 dt + \frac{1}{4\sigma}\int_0^1 \frac{\partial^3 f(x-\sigma t e_i)}{\partial x_i^3}\sigma^3(t-1)^2 dt\right| \\
&\leq \frac{\sigma^2}{4}\int_x^{x+\sigma e_i}\left|\frac{\partial^3 f(x+\sigma t e_i)}{\partial x_i^3}\right|(1-t)^2 dt + \frac{\sigma^2}{4}\int_0^1\left|\frac{\partial^3 f(x-\sigma t e_i)}{\partial x_i^3}\right|(t-1)^2 dt \\
&\leq \frac{H\sigma^2}{4}\left(-\frac{1}{3}(1-t)^3\Big|_0^1 + \frac{1}{3}(t-1)^3\Big|_0^1\right) \qquad (3.12) \\
&\leq \frac{H\sigma^2}{4}\frac{2}{3} = \frac{H\,\sigma^2}{6}. \qquad (3.13)
\end{aligned}$$

where in 3.12 we exploited the H-Lipschitz continuity of the Hessian of f.
In $\mathbb{R}^n$, remembering that $||x|| = (\sum_{i=1}^n x_i^2)^{\frac{1}{2}}$ we have:

$$||g(x) - \nabla f(x)||^2 = \sum_{i=1}^n \left(g_i(x) - \nabla f(x)^T e_i\right)^2 \leq \sum_{i=1}^n \left(\frac{H\sigma^2}{6}\right)^2 = n\left(\frac{H\sigma^2}{6}\right)^2. \quad (3.14)$$

From which derives 3.11:

$$||g(x) - \nabla f(x)|| \leq \sqrt{n}\frac{H\sigma^2}{6}.$$

□

The considerations about the difficulty of choosing the right value of the step-size σ and the fact that the optimal choice depends on the knowledge of the constant H that appears in the bound are the same discussed for the FFD scheme.

We conclude this section by noticing that the higher cost of the CFD scheme with respect to its cheaper counterpart, the FFD scheme ($2n$ function evaluations against $n+1$) is followed by a much lower bound on he estimation error. In fact, not only the denominator in the bound of the CFD scheme is three times bigger, but the estimation error goes to zero much faster as the step-size σ tends to zero – $\mathcal{O}(\sigma^2)$ insted of $\mathcal{O}(\sigma)$.

In the next section, we describe the behaviour of the CFD scheme in the presence of noise.

CFD in the presence of noise

When function evaluations are affected by noise we can write the approximation of the gradient $g(x)$ obtained with the CFD formula in the following way:

$$g(x) = \left[g_1(x), g_2(x)\ \ldots\ g_n(x)\right]^T$$

where:

$$g_i(x) = \frac{\phi(x + \sigma e_i) + \epsilon(x + \sigma e_i) - \phi(x - \sigma e_i) - \epsilon(x - \sigma e_i)}{2\sigma}. \tag{3.15}$$

With the same lines of reasoning of FFD we now analyze the estimation error in presence of noise.
Let

$$e_{\text{CFD}}(x) = g(x) - \nabla\phi(x)$$

be the estimation error and let $\{e_i\}$ be the canonical base of $\mathbb{R}^n$. We then have that:

$$E[e_{\text{CFD}}(x)] = \sum_{i=1}^{n} \frac{\phi(x + \sigma e_i) - \phi(x - \sigma e_i)}{2\sigma} e_i - \nabla\phi(x) \tag{3.16}$$

and

$$var[e_{\text{CFD}}(x)] = n\,\frac{2\lambda^2}{4\sigma^2} = \frac{n\lambda^2}{2\sigma^2} \tag{3.17}$$

If the function $f : \mathbb{R}^n \mapsto \mathbb{R}$ is twice continuously differentiable and its Hessian is H-Lipschitz continuous for all $x \in \mathbb{R}^n$ we have that

$$\|E[e_{\text{CFD}}(x)]\| \leq \sqrt{n}\,\frac{H\,\sigma^2}{6}.$$

Therefore as the step-size $\sigma \to 0$, the error goes to zero *on average* as $\mathcal{O}(\sigma^2)$ - thus faster than FFD , but its variance increases without bound as $\mathcal{O}(1/\sigma^2)$, just like FFD. There is again the trade-off between precision and accuracy that depends on the difficult choice of the parameter σ.

We provide a non-deterministic bound on the estimation error when function evaluations are affected by noise, similar to what was done in the FFD section. The following proposition is true:

Proposition 3.1.4. *If the function $f : \mathbb{R}^n \mapsto \mathbb{R}$ is twice continuously differentiable and its Hessian is H-Lipschitz continuous for all $x \in \mathbb{R}^n$, under the assumption of function evaluations are affected by noise as modelled in (3.1) we have that, with probability α:*

$$||g(x) - \nabla\phi(x)|| \leq \sqrt{n}\frac{H\sigma^2}{6} + \mathcal{P}(\alpha)\sqrt{\frac{n}{2}}\,\frac{\lambda}{\sigma}. \tag{3.18}$$

where $g(x) = \sum_{i=1}^{n} g_i(x)$ and $\mathcal{P}(\alpha)$ is the α-quantile of the standard gaussian distribution.

The proof can be derived directly from the one of Prop. 3.1.3 and basic statistical theory and is not reported here for brevity.

This proposition concludes the section on the Finite Difference scheme, one of the most well-known and widely used gradient estimation schemes due to its simplicity and high accuracy. Its importance stems from the fact that it allows for deterministic

bounds on the approximation error between the gradient estimate and its real value. The presence of noise has been shown to affect this scheme on several levels: it leads to higher bounds on the estimation error, makes choosing the step-size σ more difficult, and only allows for a statistical characterization of the estimation error.

In the following section we describe another popular scheme for approximating the gradient of a function: the Gaussian Smoothing scheme.

3.1.2 Gaussian Smoothing scheme

Another popular method for approximating an unknown gradient using only function values is the Gaussian Smoothing scheme, often used when function evaluations are affected by noise [85] and in several recent papers [117, 127, 136, 145].

Gaussian smoothing is defined as the convolution of the function $f : \mathbb{R}^n \mapsto \mathbb{R}$ with a kernel $\varphi(s) : \mathbb{R}^n \mapsto \mathbb{R}$

$$G_\sigma(x) := \frac{1}{\sigma} \int_{\mathbb{R}^n} f(x + \sigma s)\, s\, \varphi(s)\, ds \tag{3.19}$$

where $\varphi(s)$ is the probability density function (pdf) for the multivariate normal distribution with mean 0 and covariance matrix $\Sigma = I$, i.e.

$$\varphi(s) \sim \mathcal{N}(0, I_n) = \frac{1}{(\sqrt{2\pi})^n} \exp\{-\frac{1}{2} \sum_{i=1}^{n} {s_i}^2\} = \prod_{i=1}^{n} \varphi(s_i) \tag{3.20}$$

This convolution can be seen as a smoothed version of the underlying function f, where the degree of smoothing depends on the smoothing parameter $\sigma > 0$.
We will now work out formula (3.19) to provide a better understanding of this method.

Let us consider this further notation: for any $x \in \mathbb{R}^n$ denote by $\bar{x}_i \in \mathbb{R}^{n-1}$ the following vector $\Big[x_1, x_2 \ldots, x_{i-1}, x_{i+1}, \ldots, x_n\Big]^T$. With some abuse of notation, but for sake of simplicity in the use of formulas, when addressing a given coordinate x_i in a vector x let us write x as $[x_i \bar{x}_i]^T$ and denote $f(x)$ as $f(x_i, \bar{x}_i)$ and $\varphi(s) = \varphi(s_i)\varphi(\bar{s}_i)$, with $\varphi(\bar{s}_i) = \prod_{j \neq i}^{n} \varphi(s_j)$; consistently, the volume element becomes $ds = ds_i \cdot d\bar{s}_i$. In case of a vector function $f(z)$, to address explicitly its $i-th$ entry we write it as $[(f(z))_i\ \overline{(f(z))}_i]^T$.
Then, estimate (3.19) can be rewritten as follows

$$G_\sigma(x) = \frac{1}{\sigma} \int_{\mathbb{R}^n} f(x_1 + \sigma s_1, \ldots, x_n + \sigma s_n) \begin{pmatrix} s_1 \\ \vdots \\ s_n \end{pmatrix} \prod_{i=1}^{n} \varphi(s_i)\, ds \tag{3.21}$$

$$= \begin{bmatrix} \frac{1}{\sigma} \int_{\mathbb{R}^n} f(x_1 + \sigma s_1, \bar{x}_1 + \sigma \bar{s}_1)\, s_1\, \varphi(s_1)\varphi(\bar{s}_1)\, ds_1\, d\bar{s}_1 \\ \vdots \\ \frac{1}{\sigma} \int_{\mathbb{R}^n} f(x_i + \sigma s_i, \bar{x}_i + \sigma \bar{s}_i)\, s_i\, \varphi(s_i)\varphi(\bar{s}_i)\, ds_i\, d\bar{s}_i \\ \vdots \\ \frac{1}{\sigma} \int_{\mathbb{R}^n} f(x_n + \sigma s_n, \bar{x}_n + \sigma \bar{s}_n)\, s_n\, \varphi(s_n)\varphi(\bar{s}_n)\, ds_n\, d\bar{s}_n \end{bmatrix} \tag{3.22}$$

Let us consider the generic entry of vector (3.22)

$$(G_\sigma(x))_i = \frac{1}{\sigma}\int_{\mathbb{R}^n} f(x_i + \sigma s_i, \bar{x}_i + \sigma\bar{s}_i)s_i\varphi(s_i)\varphi(\bar{s}_i)ds_i\, d\bar{s}_i \tag{3.23}$$

By the Fubini theorem we can compute it as follows

$$(G_\sigma(x))_i = \int_{\mathbb{R}^{n-1}} \varphi(\bar{s}_i)\left(\frac{1}{\sigma}\int_{-\infty}^{+\infty} f(x_i + \sigma s_i, \bar{x}_i + \sigma\bar{s}_i)s_i\varphi(s_i)ds_i\right)d\bar{s}_i \tag{3.24}$$

The expression in parentheses is the estimate of the directional derivative of $f(x)$ along the i-th coordinate x_i and computed at the point $(x_i, \bar{x}_i + \sigma\bar{s}_i)$, i.e.

$$g_\sigma(x_i, \bar{x}_i + \sigma\bar{s}_i) := \frac{1}{\sigma}\int_{-\infty}^{+\infty} f(x_i + \sigma s_i, \bar{x}_i + \sigma\bar{s}_i)s_i\varphi(s_i)ds_i. \tag{3.25}$$

Hence expression (3.23) becomes

$$(G_\sigma(x))_i = \frac{1}{\sigma}\int_{\mathbb{R}^{n-1}} g_\sigma(x_i, \bar{x}_i + \sigma\bar{s}_i)\,\varphi(\bar{s}_i)\,d\bar{s}_i. \tag{3.26}$$

Therefore, the generic entry of the gradient estimate $G_\sigma(x)$ in formula (3.22) is the average of function (3.25) weighted by a $(n-1)$-dimensional Gaussian kernel $\varphi(\bar{s}_i) = \mathcal{N}(0, I_{n-1})$ over the subspace $\mathbb{R}^{n-1}$ of $\mathbb{R}^n$. As a consequence, the computation of any entry of vector $G_\sigma(x)$ implies an integration over $\mathbb{R}^n$, that clearly represents something not easy to compute.

This problem is overcome in Gaussian Smoothing methods, as in papers [12, 16] by considering that (3.19) is indeed an ensemble average of function $f(x + \sigma s)s$ over all the directions $s \in \mathbb{R}^n$ weighted by the Gaussian distribution $\varphi(s) \sim \mathcal{N}(0, I_n)$. It is in fact possible to rewrite (3.19) in the following way:

$$G_\sigma(x) = \frac{1}{\sigma}\,\mathbb{E}_\varphi[f(x + \sigma s)s]. \tag{3.27}$$

From this expression, it is possible to derive some bounds for the distance between $G_\sigma(x)$ and $\nabla f(x)$, depending on the assumptions on the function f. If the gradient of the function f is L-Lipschitz continuous, we have that

$$||G_\sigma(x) - \nabla f(x)|| \leq \sqrt{n}\,L\,\sigma. \tag{3.28}$$

Instead, if the Hessian of the function f is H-Lipschitz continuous, we have that

$$||G_\sigma(x) - \nabla f(x)|| \leq \sqrt{n}\,H\,\sigma^2. \tag{3.29}$$

Proofs of (3.28) and (3.28) can be found in [17], for the special case of absence of noise.

From [17] we also report the bounds on the distance between $G_\sigma(x)$ and $\nabla f(x)$ in presence of an additive noise and most of the prepositions of the following sections. Differently from the characterization of the noise described in (3.1), the authors assume that the value of the noise is bounded above by a constant λ_f, i.e. $|\epsilon(x)| \leq \lambda_f$

for all $x \in \mathbb{R}^n$. [2] The two bounds on the error between $G_\sigma(x)$ and $\nabla f(x)$ in this setting change in the following way:
if the gradient of the function f is L-Lipschitz continuous, we have:

$$||G_\sigma(x) - \nabla\phi(x)|| \leq \sqrt{n}\, L\, \sigma + \frac{\sqrt{n}\lambda_f}{\sigma}. \tag{3.30}$$

If instead the Hessian of the function f is H-Lipschitz continuous, we have:

$$||G_\sigma(x) - \nabla\phi(x)|| \leq \sqrt{n}\, H\, \sigma^2 + \frac{\sqrt{n}\lambda_f}{\sigma}. \tag{3.31}$$

We again are facing a trade-off: the choice of the parameter σ affects the two components of this bound in the opposite way, thus representing a choice that has to be handled with care.
The interpretation of (3.19) as the ensemble average (3.27) is the proper starting point of the Gaussian Smoothing scheme, that exists in its *Gaussian Smoothing Gradient* (*GSG*) and *central Gaussian Smoothing Gradient* (*cGSG*) forms. The proofs of the propositions reported in the following sections can be found in [17].

Gaussian Smoothing Gradient

The ensemble average (3.27) can be well approximated by sampling a set of M independent directions $\{s_i\}$ in $\mathbb{R}^n$ according to $\mathcal{N}(0, I_n)$, and considering the *sample average approximation* of $E_\varphi[f(x + \sigma s)s]$. In the GSG method, the approximation $g(x)$ of the gradient is obtained by:

$$g(x) = \frac{1}{M}\sum_{i=1}^{M} \frac{(f(x + \sigma s_i) - f(x))s_i}{\sigma}. \tag{3.32}$$

This sample average is an unbiased estimator of $\nabla f(x)$ and its accuracy increases with increasing M. We can see that the approach of this method is somewhat similar to that of the finite difference scheme with two major differences:

- the perturbation of the point x occurs along directions sampled from a Gaussian distribution in the GSG method, and not along the canonical vector as in the finite difference scheme. (3.32) can in fact be interpreted as the average of several forward finite difference versions of the directional derivative of f at x along s_i [98].
- The number of direction sampled, and consequently of function evaluations, in the GSG method is independent of the dimension n of the function f: $M + 1$ function computations in case of (3.32), whilst you need exactly $n + 1$ function evaluations for the FFD scheme.

[2] Let us note that modeling the noise as in (3.1) we can obtain the same bounds (3.28) and (3.29) in a probabilistic (instead of deterministic) fashion.

In [98] the analysis of the properties of (3.19) and (3.32) can be found for the special case $M = 1$. In [17] the analysis is extended for the case of $M > 1$ and bounds on the variance of $g(x)$ and on the distance between $g(x)$ and $\nabla f(x)$ are derived.

The variance of (3.32) can be written as:

$$Var\{g(x)\} = \frac{1}{M}\,\mathbb{E}_{s\sim\mathcal{N}(0,I)}\left[\left(\frac{f(x+\sigma s)-f(x)}{\sigma}\right)^2 s\,s^T\right] - \frac{1}{M}G_\sigma(x)G_\sigma(x)^T,$$

and the following proposition holds:

Proposition 3.1.5. *Being $g(x)$ the gradient estimate obtained according to (3.32), if the function $f : \mathbb{R}^n \mapsto \mathbb{R}$ is continuously differentiable and its Gradient is L-Lipschitz continuous for all $x \in \mathbb{R}^n$ we have that:*

$$Var\{g(x)\} \preceq \kappa(x)I, \;\; \textit{where } \kappa(x) = \frac{36||\nabla f(x)||^2 + 3L^2\sigma^2(n+2)(n+4)}{4M}.$$

The analysis of the distance between $g(x)$ and $\nabla f(x)$ for the GSG method is trickier than that of FFD. Because of the randomness introduced by the random sampling of Gaussian directions u_i, the error can only be described statistically, as shown in the following theorem:

Theorem 3.1.6. *If the function $f : \mathbb{R}^n \mapsto \mathbb{R}$ is continuously differentiable with a L-Lipschitz continuous Gradient for all $x \in \mathbb{R}^n$, if $g(x)$ is the gradient estimate obtained according to (3.32) with sample size:*

$$M \geq \frac{9n||\nabla f(x)||^2}{\delta\, r^2} + \frac{3n(n+2)(n+4)\,L^2\,\sigma^2}{4\,\delta\, r^2}$$

then, for all $x \in \mathbb{R}^n$ and any $r > 0$, $||g(x) - \nabla f(x)|| \leq \sqrt{n}\,L\,\sigma + r$ with probability at least $1-\delta$, with $0 < \delta < 1$.

Despite the fact that this is an intriguing result, it is difficult to put into practice. In fact, to calculate the sample size needed to obtain the desired bound on the norm of distance between the gradient approximation and the real gradient, one should know two information rarely available: the value of the norm of the gradient in the point of interest, and the value of the Lipschitz constant. Furthermore, the number of function evaluations required to obtain the desired bound is likely to be quite large.

In the following section, we show how Prop. 3.1.5 and Thm. 3.1.6 change if the sampling of the function is affected by additive noise.

GSG in the presence of noise

If an additive noise affects the sampled function values f(x), as in 3.1, assuming that the value of the noise is bounded above by a constant λ_f, i.e. $|\epsilon(x)| \leq \lambda_f$ for all $x \in \mathbb{R}^n$, the following proposition holds:

Proposition 3.1.7. *Being $g(x)$ the gradient estimate obtained according to (3.32), if the function $\phi : \mathbb{R}^n \mapsto \mathbb{R}$ is continuously differentiable and its Gradient is L-Lipschitz continuous for all $x \in \mathbb{R}^n$ we have that:*

$$Var\{g(x)\} \preceq \kappa(x)I, \text{ where } \kappa(x) = \frac{36||\nabla\phi(x)||^2 + 3L^2\sigma^2(n+2)(n+4)}{4M} + \frac{12\lambda_f}{M\sigma^2}.$$

Theorem 3.1.8. *If the function $\phi : \mathbb{R}^n \mapsto \mathbb{R}$ is continuously differentiable with a L-Lipschitz continuous Gradient for all $x \in \mathbb{R}^n$, if $g(x)$ is the gradient estimate obtained according to (3.32) with sample size:*

$$M \geq \frac{9n||\nabla\phi(x)||^2}{\delta\, r^2} + \frac{3n(n+2)(n+4)\, L^2\, \sigma^2}{4\,\delta\, r^2} + \frac{12n\lambda_f^2}{\delta\, r^2\sigma^2}$$

then, for all $x \in \mathbb{R}^n$ and any $r > 0$, $||g(x) - \nabla\phi(x)|| \leq \sqrt{n}\, L\, \sigma + r$ with probability at least $1 - \delta$, with $0 < \delta < 1$.

With respect to the noise-free setting, we can notice that both the upper bound on the variance and and the lower bound on M (the sufficient sample size for guaranteeing a high accuracy in the estimate) have one more term that is directly proportionate to the bound on the maximum value of the noise and inversely proportionate to the step-size σ.

Both Prop. 3.1.7 and Thm. 3.1.8 highlight again the difficulty of choosing the right value of σ, that becomes even harder when function evaluations are affected by noise, as described before for other approximation schemes.

In the following section we describe the central Gaussian Smoothing Gradient (cGSG), a more accurate version of the GSG scheme that we can think of as the equivalent of the CFD scheme to its FFD counterpart.

Central Gaussian Smoothing Gradient

Similarly to the Finite Difference scheme, there is also a more accurate version of the Gaussian Smoothing Gradient that requires more function evaluations per sampled direction: its cGSG form, that computes the approximation $g(x)$ of the gradient in the following way:

$$g(x) = \frac{1}{M}\sum_{i=1}^{M} \frac{(f(x + \sigma s_i) - f(x - \sigma s_i))s_i}{2\sigma}. \tag{3.33}$$

This sample average is also an unbiased estimator of $\nabla f(x)$ and its accuracy increases with increasing M. The computation of (3.33) requires $2M$ function evaluations, where again M is independent of the dimension n of the function, against the $2n$ function evaluations required by the CFD method.
Similarly to the GSG section, we report now the variance of this method and its bound, along with a theorem that links the sample size to the desired bound on the approximation error.

The variance of (3.33) can be written as:

$$Var\{g(x)\} = \frac{1}{M}\,\mathbb{E}_{s\sim\mathcal{N}(0,I)}\left[\left(\frac{f(x+\sigma s)-f(x-\sigma s)}{2\sigma}\right)^2 s\,s^T\right] - \frac{1}{M}G_\sigma(x)G_\sigma(x)^T.$$

and the following proposition holds:

Proposition 3.1.9. *Being $g(x)$ the gradient estimate obtained according to (3.33), if the function $f:\mathbb{R}^n\mapsto\mathbb{R}$ is twice continuously differentiable and its Hessian is H-Lipschitz continuous for all $x\in\mathbb{R}^n$ we have that:*

$$Var\{g(x)\} \preceq \kappa(x)I,\ \ where\kappa(x) = \frac{9||\nabla f(x)||^2}{M} + \frac{H^2\sigma^4(n+2)(n+4)(n+6)}{12M}. \tag{3.34}$$

At this point we can report the analogous for the cGSG of Thm. 3.1.6, where the sample size is linked to the desired bound on the norm between the approximation of the gradient and the real one.

Theorem 3.1.10. *If the function $f:\mathbb{R}^n\mapsto\mathbb{R}$ is twice continuously differentiable with a H-Lipschitz continuous Hessian for all $x\in\mathbb{R}^n$, if $g(x)$ is the gradient estimate obtained according to (3.33) with sample size 2M where*

$$M \geq \frac{9n||\nabla f(x)||^2}{\delta\, r^2} + \frac{n(n+2)(n+4)(n+6)\,H^2\,\sigma^4}{12\,\delta\, r^2}$$

then, for all $x\in\mathbb{R}^n$ and any $r>0$, $||g(x)-\nabla f(x)||\leq\sqrt{n}\,M\,\sigma^2+r$ with probability at least $1-\delta$ with $0<\delta<1$..

The considerations about the usefulness of this theorem from a practical point of view are the same reported after Thm. 3.1.6.

cGSG in the presence of noise

Let us now report how the results shown in Sect. 3.1.2 change when function evaluations are affected by noise, as in 3.1, with $|\epsilon(x)|\leq\lambda_f$ for all $x\in\mathbb{R}^n$.

Proposition 3.1.11. *Being $g(x)$ the gradient estimate obtained according to (3.32), if the function $\phi:\mathbb{R}^n\mapsto\mathbb{R}$ is continuously differentiable and its Gradient is L-Lipschitz continuous for all $x\in\mathbb{R}^n$ we have that:*

$$Var\{g(x)\} \preceq \kappa(x)I,\ \ where\ \kappa(x) = \frac{9||\nabla\phi(x)||^2}{M} + \frac{H^2\sigma^4(n+2)(n+4)(n+6)}{12M} + \frac{3\lambda_f}{M\sigma^2}.$$

Theorem 3.1.12. *If the function $\phi:\mathbb{R}^n\mapsto\mathbb{R}$ is continuously differentiable with a L-Lipschitz continuous Gradient for all $x\in\mathbb{R}^n$, if $g(x)$ is the gradient estimate obtained according to (3.32) with sample size:*

$$M \geq \frac{9n||\nabla\phi(x)||^2}{\delta\, r^2} + \frac{n(n+2)(n+4)(n+6)\,H^2\,\sigma^4}{12\,\delta\, r^2} + \frac{3\lambda_f^2}{\delta r^2\sigma^2} \tag{3.35}$$

then, for all $x\in\mathbb{R}^n$ and any $r>0$, $||g(x)-\nabla\phi(x)||\leq\sqrt{n}\,L\,\sigma+r$ with probability at least $1-\delta$, with $0<\delta<1$.

This theorem concludes the section on the Gaussian Smoothing Gradient scheme, another popular method for approximating the gradient using only function values.

Compared with the Finite Difference scheme, it has both some advantages and disadvantages. Its main advantage is the fact that the number of function evaluations required to compute the gradient approximation is *independent* of the number of variables n. In fact, while the Finite Difference scheme requires either $n + 1$ or $2n$ function evaluations to build the gradient estimate, the Gaussian Smoothing Gradient scheme requires a minimum of two function evaluations to build a gradient estimate, e.g. by using $M = 1$ in (3.32). It is worth noting that this is rarely - if ever - done in practice, because the gradient estimate produced with such a small number of function evaluations would be inaccurate, and that the number of function evaluations required for accurate gradient approximation is frequently higher than n as shown in the numerical experiments in the following chapter.

The other advantage of this scheme is the filtering effect of the convolution between the objective function f and the Gaussian Kernel, that makes the scheme a little more robust to the noise affecting function evaluations. The interpretation of the integral (3.19) as the ensemble average (3.27) and its subsequent sample average approximation, implies that this robustness to the noise diminishes as the precision of the approximation decreases.

As for the downsides, the Gaussian Smoothing scheme only allows for a statistical characterization of the estimation error *also* in the noise free setting. The reason for this lies in the randomicity that comes from the random sampling of the directions from the gaussian distribution.

The other downside is the poor accuracy of this scheme with respect to the finite difference scheme in the noise-free setting, when both methods use the same number of function evaluations. Because of the convoluted way the bounds on the estimation error of the gaussian smoothing scheme are expressed, it is hard to justify this claim on a theoretical level. However, numerical results in the following chapter will highlight this downside.

We conclude this section by briefly describing other gradient approximation methods, before moving to the next chapter where we describe a new proposed scheme for building gradient approximations.

3.1.3 Brief outline of other methods

Despite being the most popular, the finite difference scheme and the Gaussian smoothing scheme are not the only existing gradient approximation methods.

Another method that is conceptually similar to the Gaussian smoothing scheme is the one proposed by A. Flaxman in [47], where the function f is smoothed with a uniform distribution on a ball, as in:

$$G_\sigma(x) := \frac{n}{\sigma}\,\mathbb{E}_{s\sim\mathcal{U}(\Sigma(0,1)}[f(x+\sigma s)s]. \tag{3.36}$$

where $\Sigma(0,1)$ represents a unit sphere of radius 1 centered at 0. Similarly to the Gaussian Smoothing case, the ensemble average (3.36) can be approximated with the following sample average approximation:

$$g(x) = \frac{n}{M}\sum_{i=1}^{M}\frac{f(x+\sigma s_i)-f(x)}{\sigma}\,s_i \tag{3.37}$$

where the vectors u_i are M independently and identically distributed random vectors following a uniform distribution on the unit sphere. Using twice the number of function evaluations, it is also possible to approximate (3.36) with the following:

$$g(x) = \frac{n}{M}\sum_{i=1}^{M}\frac{f(x+\sigma s_i)-f(x-\sigma s_i)}{\sigma}\,s_i. \tag{3.38}$$

In [17] it is possible to find bounds on the variance of this estimate and the lower bound on the number of sampled direction M sufficient for guaranteeing the desired accuracy with a certain probability δ, similar to those shown in Sect. 3.1.2. In particular, the authors show an advantage that this method has over gaussian smoothing, namely that when deriving the lower bound on M the structure of this method allows the use of Bernstein's inequality instead of Chebychev's, thus resulting in a bound with an improved dependence on the probability δ.

This method is similar to the estimate proposed by Spall [124] and Granich [54] in two different, independent works, where a perturbation vector p in which every component is an independent random variable with zero mean, is used to build the gradient estimate according to the following formula:

$$g(x) = \frac{f(x+\sigma p)}{\sigma}\left[\frac{1}{p_1},\frac{1}{p_2},\cdots\frac{1}{p_n}\right]^T. \tag{3.39}$$

Another interesting method for producing gradient estimates is the one proposed in [16], built around the concept of interpolation models that are popular in the world of Derivative Free Optimization (see for example [31, 109, 137]). The authors focus on the simplest interpolation model, the linear one, around a point $x \in R^n$, as in:

$$m(y) = f(x) + g(x)^T(y-x). \tag{3.40}$$

They then consider the directions $u_i,\ \ i = 1,\ldots,n$ and define the matrix $Q_\chi \in \mathbb{R}^{nxn}$ as the matrix with rows u_i. Defining the vector $F_\chi \in \mathbb{R}^n$ as the vector with entries $f(x+\sigma u_i) - f(x)$ for $i = 1,\ldots,n$, the linear interpolation model satisfying the interpolation condition $f(x+\sigma u_i) = m(x+\sigma u_i)$ for each direction can be written as:

$$\sigma Q_\chi g(x) = F_\chi. \tag{3.41}$$

If the determinant of the matrix Q_χ is not zero, it is possible to write $g(x) = \frac{1}{\sigma} Q_\chi^{-1} F_\chi$. When the directions u_i are orthonormal, $Q_\chi^{-1} = Q_\chi^T$ and the gradient estimate $g(x)$ can be written as:

$$g(x) = \sum_{i=1}^{n} \frac{f(x + \sigma u_i) - f(x)}{\sigma} u_i. \tag{3.42}$$

We can notice the similarities between estimate (3.42) and (3.32), where the former can be interpreted as a scaled version of the latter, the difference being in the expected length of the sampled directions when they are orthonormal and when they come from a Gaussian distribution.

This gradient estimate method concludes this section, where we described several gradient approximation schemes, with a special focus on the Finite Difference scheme and the Gaussian Smoothing one. We described their origins and described some of their properties, both in the noise-free and in the noisy setting. In the next chapter we describe the origin and the properties of a new scheme for approximating the gradient of a function, and we show how it performs in comparison to the main methods described in this section.

3.2 Introduction to neural networks

Estimating the gradient of a function at each iteration of any algorithm in charge of the resolution of a minimization problem would be hardly feasible in a reasonable time when the evaluations of the objective function are very expensive in terms of time and resources, such as when they require the use of an external simulation software. In this scenario, it might be wiser to use function evaluations to directly create a *metamodel* (or *surrogate* model) that approximates the objective function.

There are a variety of approaches that can be used to build a metamodel, including but not limited to:

- smoothing splines [55];
- radial basis functions [25];
- Kriging approximators, [105];
- neural networks.

As far as the authors know, there are no results in the literature that support the choice of one approach over the others and, since the objective is that of studying the feasibility of the metamodel creation, the in-depth analysis about which approach works better is left to future studies.

In this section, and in the rest of this thesis, we focus on the employment of neural networks for the creation of the approximating model, given the growing

amount of interest around this topic and the availability of several open source software.

The objective of this section is to introduce some basic concepts about neural networks. In particular, after describing the simple Perceptron, we introduce the class of Feedforward Neural Networks (FNN) known as Multi-Layer Perceptron (MLP), and we report some results on their *approximation capabilities*. We also introduce the problem of *training* a neural network, whose details will be provided in the case study reported in Chapter 5.

We begin by introducing the concept of the *neural neuron*, also referred to simply as *neuron*. The neuron, originally presented in 1943 by McCulloch e Pitts [87] and later on by Rosenblatt [113], allegedly simulates the behaviour of the human nervous system. It is a mathematical function that takes as input a generic vector $x^p \in \mathbb{R}^n$, performs a nonlinear transformation of it and returns a scalar *output* $\bar{y}^p$. For the analogy between the artificial neurons and the biological ones see [2]. The nonlinear transformation consists of the application of a nonlinear function $g(\cdot)$ to the sum of a value $b \in \mathbb{R}$ known as *bias term* and the scalar product between the input vector x^p and the vector $w \in \mathbb{R}^n$, known as *weights vector*, as in equation (3.43)

$$\bar{y}^p = g(w^T x^p + b) = g\left(\sum_{i=1}^{n} w_i x_i + b\right). \tag{3.43}$$

Function $g(\cdot)$ in (3.43) is known as *activation function*. Thus, activation functions are applied by the neuron in order to introduce non-linearities in the model. From the introduction of the Perceptron throughout the incredible development of the Machine Learning field until today, several dozens of activation functions have been proposed. In Appendix A the reader can find a description of some of the most popular ones.

3.2.1 Perceptron

The *neuron* is the central element of the *single-layer Perceptron*, also known as *Perceptron*. It consists of two layers (i.e. $L = 2$) of nodes:

- n input nodes that are used to transmit input values to the output node and don't perform any type of computation.
- the output node, or *neuron*, described above.

A visualization of the Perceptron is reported in Fig. 3.2.

The interpretation of the Perceptron proposed by Rosenblatt is quite interesting since he considers the Perceptron a *linear classifier*. Indeed, given a set of inputs

$$\{x^p, y^p\}_{p=1}^{P}, \quad x^p \in \mathbb{R}^n, \quad y^p \in \{-1, +1\},$$

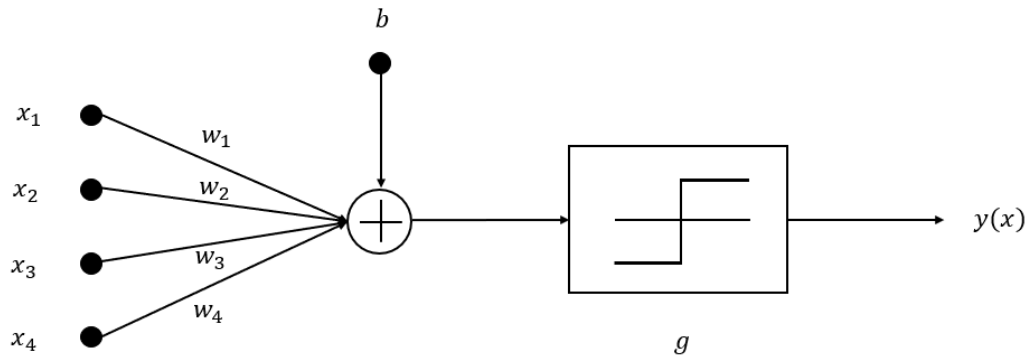

Figure 3.2. Scheme of a Perceptron with P = 4

the output $\bar{y}^p$ of the Perceptron for a given input x^p is either -1 or $+1$ depending on the sign of $w^T x^p + b$.
The objective will therefore be that of finding the values of (w, b) such that

$$w^T x^p + b \geq 0 \quad \text{if} \quad y^p = +1$$
$$w^T x^p + b < 0 \quad \text{if} \quad y^p = -1$$

and these values can be found by performing a *training process*.

Furthermore, note that finding the values of w and b that satisfy the equations above is possible only if the set of input $\{x^p, y^p\}_{p=1}^P$ is *linearly separable*. This fact helps thinking at the geometrical interpretation of the Perceptron as a separating hyperplane

$$H = \{x \in R^n : \quad w^T x + b = 0\}$$

that strictly separates the sets

$$A = \{x^p : (x^p, y^p) \in T, y^p = +1\} \quad \text{and} \quad B = \{x^p : (x^p, y^p) \in T, y^p = -1\}.$$

To summarize, the Perceptron is the most basic type of a neural network, with only one input and one output layer. The function learnt by the Perceptron is extremely simple, and not sufficient for the majority of applications, being used in practice only for linear separation problems [107]. As a result, multi-layer neural networks were developed, which are more complex models that can be thought of as an evolution or extension of the simple Perceptron. One of these more complicated models is the MLP.

3.2.2 Multi-Layer Perceptron

As suggested by the name, MLP is characterized by the presence of $L \geq 2$ layers. The neural units, or *neurons*, are organized in them in this fashion:

- the input nodes connects the input variables $(x^p)_i \in \mathbb{R}, \quad i = 1, \ldots, n$ to the first layer without performing any computation;
- the first L - 1 layers are called *hidden layers*, and neurons belonging to them don't have any direct connection with the output;
- the last layer is called *output layer*, and is formed by K different neural units, each of which represents the output $\bar{y}_i \in \mathbb{R}, \quad i = 1, \ldots, K$ of the network.
- each neuron in a given layer l is connected by weighted arcs to the outputs of each neurons of the layer $l - 1$. Conversely, the output of each neuron in a layer l is connected to all the neurons of the layer $l + 1$. Finally, there are no connections between neurons of the same layer. The characteristic of having connections between *all* the neurons of adjacent layers makes the MLP a *fully connected* architecture. There are also types of artificial neural networks where this does not happen, or where other type of connections - such as those between the output of a neuron and the neuron itself - exist, like in the case of *Recurrent Neural Networks* (RNN).

An example of the MLP, that can be also seen as a directional acyclical graph where nodes represent neural units and edges represent connections between neurons, is represented in Fig. 3.3.

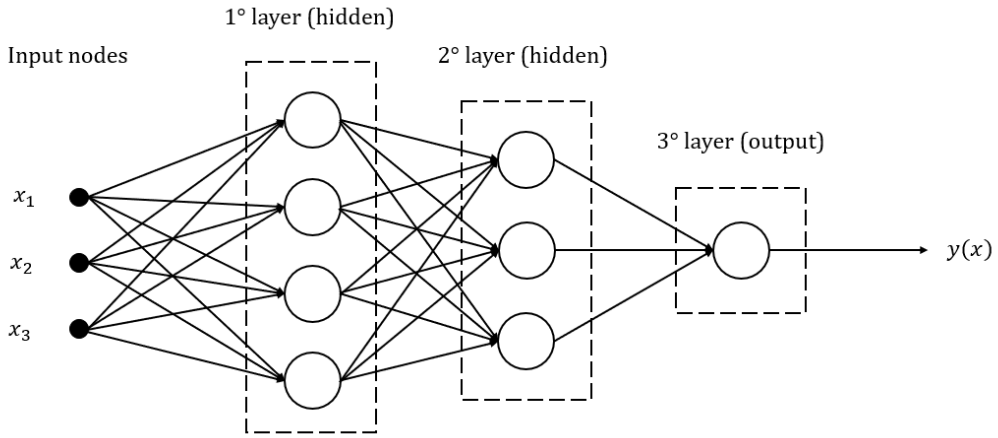

Figure 3.3. Scheme of a MLP with P = 3, four neurons in the first hidden layer, three neurons in the second and one neuron in the output layer

As indicated by this figure and the description above, the generic neuron k of the layer l ($l > 1$) produces the output z_j^l in the following way

$$z_j^l = g_j^l(a_j^l)$$

where

$$a_j^l = \sum_{i=1}^{N^{l-1}} w_{ji}^l z_i^{l-1} + b_j^l, \qquad z_j^{l-1} = g_j^{l-1}(a_j^{l-1}). \tag{3.44}$$

and N^l denote the number of neurons of the layer l.
Therefore, the expression of the output computed by the first layer is the following

$$z_j^1 = g_j^1(a_j^1), \quad a_j^1 = \sum_{i=1}^{n} w_{ji}^1 x_i + b_j^1, \tag{3.45}$$

whereas for $l = L$ we have

$$N^L = K \text{ and } z_j^L = a_j^L := \bar{y}_j, \quad j = 1, \ldots, K.$$

3.2.3 Neural networks as universal approximators

To simplify the analysis we refer in this section to a MLP characterized by one-dimensional output (K = 1), one hidden layer and the same activation function $g(\cdot)$ for all the neurons of the hidden layer. We also assume that the output neuron does not apply any nonlinear transformation to its computation, i.e. there is no activation function on the output neuron.
Let us also introduce the following notation:

- N: number of neurons of the hidden layer;
- w_{ji}: weight of the arc connecting input node i with neuron j of the hidden layer layer;
- b_j: threshold of hidden neuron j;
- v_j : output weight of the arc connecting hidden neuron j to the output node;
- g: activation function of the hidden neurons;

With the notation described above, for a given input $x^p \in \mathbb{R}^n$, the two-layer MLP computes the value $\bar{y}^p(x^p)$ in the following way:

$$\bar{y}^p(x^p) = \sum_{j=1}^{N} v_j \, g\left(\sum_{i=1}^{n} w_{ji} (x^p)_i + b_j \right) \tag{3.46}$$

where $(x^p)_j$ indicates the j_{th} component of vector x^p.

Given a set of points $X = \{x^p \in \mathbb{R}^n, \ p = 1, \ldots, P\}$ and the corresponding set of scalars $y = \{y^p \in \mathbb{R}, \ p = 1, \ldots, P\}$, the *interpolation problem* is the problem of finding a function $s : \mathbb{R}^n \to \mathbb{R}$, in some given class of real functions, that satisfies the following

$$s(x^p) = y^p, \quad p = 1, \ldots, P.$$

The well known theorem that follows, shows that the function obtained with a two-layer MLP, with N neurons in the hidden layer and under appropriate assumptions on the activation function g, can solve the interpolation problem.
Let us now report this theorem from [107]:

Theorem 3.2.1. *Let g be a continuous function in $\mathbb{R}$ and assume g is not polynomial. Then, given P distinct vectors $x^p \in \mathbb{R}^n$ and P scalars $y^p \in \mathbb{R}$, for $p = 1, \dots, P$ there exist P vectors $w_j \in \mathbb{R}^n$ and $2P$ scalars $v_j, b_j \in \mathbb{R}$, for $j = 1, \dots, P$, such that:*

$$\sum_{j=1}^{N} v_j \, g\left(w_j^T x^p + b_j\right) = y^p, \quad p = 1, \dots, P \tag{3.47}$$

This theorem implies that, given a function $f \in C(X, \mathbb{R})$ defined on the compact set $X \subset \mathbb{R}^n$, for every $\epsilon > 0$ a 2-layer MLP with a non-polynomial activation function can be constructed so that:

$$|f(x) - \bar{y}(x)| < \epsilon, \text{ for all } x \in X. \tag{3.48}$$

where $\bar{y}(x)$ represents the output of the network obtained with (3.47).

Thm. 3.2.1 echoes the 1989 *universal approximation theorem* [36, 66] that states that a FFN with a linear output layer and at least one hidden layer with any "squashing" activation function (such as the logistic sigmoid activation function) can approximate any Borel measurable function from one finite-dimensional space to another with any desired nonzero amount of error, provided that the network is given enough hidden units [52].

3.2.4 The training problem

Despite being universal approximators, MLPs and neural networks in general are seldom used to solve interpolation problems. In fact, given a set of observations $\{x^p, y^p\}_{p=1}^P$ known as "training data", they are normally implemented to *learn* the underlying distribution of the data in order to create a model that can successfully predict new, unseen, data.

Taking into account equation (3.46), the aim of the training consists in finding the values of the vectors w, v, b – that will be indicated by the symbol θ that represents all the learnable parameters of the model – so that the distance between the predicted $\bar{y}^p$ computed as $f(x^p, \theta)$ and the real y^p is small.
There are different ways to measure this distance and they mainly depend on the goal of the model. At a very high level, under the umbrella of supervised learning, we can distinguish between classification and regression problems. [3] The former are problems where the output variable has discrete values, whereas in the latter the model is asked to output a function $f : \mathbb{R}^n \to \mathbb{R}$.
In the case study in Chapter 5 the focus will be on regression problems and the loss function used for the training will be one of the most popular for this class of problems: the Mean Squared Error (MSE), defined as

$$\mathcal{L}(y^p, f(x^p, \theta)) = (f(x^p, \theta) - y^p)^2. \tag{3.49}$$

[3] For a description of the differences between supervised and unsupervised learning, and on the details about classification problems see Chapter 5 in [52].

The training of an MLP requires solving the following optimization problem:

$$\min_{\theta} \frac{1}{P} \sum_{p=1}^{P} \mathcal{L}(y^p, f(x^p, \theta)) + \rho||\theta||^2 \tag{3.50}$$

where the second term represents a regularization term, with $\rho > 0$ being a hyperparameter whose value needs to be set.

The optimization problem (3.50) is unconstrained and bounded below by zero and its regularization term is important as it makes the function coercive, thus guaranteeing the existence of at least one optimal solution.
Solving this optimization problem requires the computation of the gradient of the objective function. Because of the potential multi-layer structure of the network, this computation is not easy and the difficulty of solving the optimization problem is the reason behind the limited research around multi-layer networks until 1985. This is the year when *Back-Propagation* algorithm, that is still the algorithm used for training ML models today, was proposed by Rumelhart et al. [114, 115]. It is an algorithm that updates the gradient of the error function with respect to the neural network's weights. The name "backwards" is related to the efficient calculation of the gradient that proceeds from the deepest to the shallowest layer, where the gradient computations from one layer are reused in the gradient computation for the previous layer.

Problem (3.50) can be seen as a realization of the *Empirical Risk Minimization* inductive principle (ERM principle) for the particular choice of the loss function (3.49). Learning problems can in fact be interpreted as a particular case of the general problem of minimizing the risk functional on the basis of empirical data. For details on the problem of risk minimization and its link with the training of ML models we recommend the renowned book published from Vapnik in 1999 [132]. The relationship between problem (3.50) and the theory of Risk Minimization not only is interesting from a knowledge point of view, but it also has some practical consequences. Differently from what happens with "classic" optimization problems, where the goal is that of finding the global minimum, the ultimate interest of training a machine learning model is to *learn* how to predict unseen data. The difference between a learning process and an optimization process stands in the purpose: optimizing the ML model until the convergence to a global minimum is not beneficial, as the network tends to memorize the data seen during the training phase, thus predicting them extremely accurately, whereas the focus should be on the ability of network to generalize. Indeed, the capacity of the model to perform well on inputs not seen during the training phase is called *generalization* and it provides insight into how well the model will perform when applied in the real world.

As a consequence of this, although the training of the model is performed using a set of data called *training set* and by minimizing the loss function on this set, a bigger interest is placed on the performance measures that are normally evaluated on a different set of data, called *test set*.

In order for the test error to be useful for predicting the behaviour of the network on new data, training and test data must satisfy the so called *i.i.d. assumptions*, which require that the samples in the dataset are independent from each other and the training set and test set are identically distributed and drawn from the same probability distribution.

Since the final parameters of the model are found by minimizing the loss function of the training data, it is reasonable to expect the test error to be greater or equal than the training error if the i.i.d. assumptions are satisfied. The relation between the two errors is of crucial importance in machine learning applications. Ideally, one should try to obtain a small training error whilst keeping the gap between the training and test error small. Two terms are often used when this does not happen:

- *underfitting* indicates that the model is not able to obtain a sufficiently small error on the training set (i.e. has an high empirical risk and will predict poorly on new data);
- *overfitting* is used when the model does not have generalization capabilities, i.e. the gap between the training and test error is too big and the model will perform poorly on unseen data [63]. For some strategies on how to avoid overfitting see [119, 125].

We have now all the elements to describe at a very high level the procedure for training a Multi-Layer Perceptron as the successive application of the following operations:

1. Data set preparation: it is needed to reduce biases in the distribution of input data and consists in the creation of the training and test sets;
2. Choice of the metrics to evaluate the performance of the trained model;
3. Definition of the architecture of the network: choice of the number of layers, the number of neurons in each layer and of eventual other hyperparameters (e.g. *dropout rate* [125]);
4. Identification of the parameters for the training procedure: choice of the loss function to minimize and of the optimization algorithm and its eventual parameters (e.g. *learning rate* and *momentum* for the Stochastic Gradient Descent (SGD) algorithm [126]).
5. Training of the network.

In Chapter 5, we consider a case study about a simulation-based optimization problem in which we replace the objective function with a metamodel obtained with an artificial neural network. In Sect. 5.4, in particular, we provide more details about the steps (1-5) described above.

Chapter 4

A new scheme for Gradient Approximation

In this Chapter, we present the analysis that led to the development of a new scheme for approximating the gradient, that we named *Normalized Mixed Finite Difference Scheme*, or *NMXFD*.

Two main motivations were behind the exploration of a new gradient approximation scheme: on one side we were looking for a scheme that would exploit the filtering power of the kernel that appears in the linear functionals commonly used for building gradient approximations. On the other hand we wanted to derive an approximation scheme for linear functionals approximating the gradient whose error of approximation could be characterized by a deterministic point of view in the case of noise-free data, and not in a stochastic way as in the case of the schemes presented in Sect. 3.1.2.

The Chapter is organized as follows: In Sect. 4.1 we explain the reasoning that led to the development of the new scheme. We propose the new gradient approximation in Sect. 4.2 and we show its theoretical properties. Numerical experiments are covered in two sections: Sect. 4.3 presents the *point-wise* comparison between *NMXFD* and other gradient approximation methods, whereas in Sect. 4.4, the estimate of the gradient obtained with the various schemes is used inside unconstrained optimization algorithms when function evaluations are affected by noise. Finally, in Sect. 4.5 we discuss the main points of the Chapter and we consider potential future works.

4.1 Origin of the new scheme

The proposed scheme has its roots in integral 3.19:

$$G_\sigma(x) := \frac{1}{\sigma} \int_{\mathbb{R}^n} f(x + \sigma s)\, s\, \varphi(s)\, \mathrm{d}s.$$

that has the following property

$$\lim_{\sigma \to 0} G_\sigma(x) = \nabla f(x).$$

as we can see from (3.28) and (3.29).
In Sect. 3.1.2 we showed that the integral can be rewritten in the following way:

$$G_\sigma(x) = \frac{1}{\sigma}\int_{\mathbb{R}^n} f(x_1+\sigma s_1,\dots,x_n+\sigma s_n)\begin{pmatrix} s_1 \\ \vdots \\ s_n \end{pmatrix}\prod_{i=1}^{n}\varphi(s_i)\,\mathrm{d}s$$

$$= \begin{bmatrix} \frac{1}{\sigma}\int_{\mathbb{R}^n} f(x_1+\sigma s_1,\bar{x}_1+\sigma\bar{s}_1)\,s_1\,\varphi(s_1)\varphi(\bar{s}_1)\,\mathrm{d}s_1\,d\bar{s}_1 \\ \vdots \\ \frac{1}{\sigma}\int_{\mathbb{R}^n} f(x_i+\sigma s_i,\bar{x}_i+\sigma\bar{s}_i)\,s_i\,\varphi(s_i)\varphi(\bar{s}_i)\,\mathrm{d}s_i\,d\bar{s}_i \\ \vdots \\ \frac{1}{\sigma}\int_{\mathbb{R}^n} f(x_n+\sigma s_n,\bar{x}_n+\sigma\bar{s}_n)\,s_n\,\varphi(s_n)\varphi(\bar{s}_n)\,\mathrm{d}s_n\,d\bar{s}_n \end{bmatrix}$$

and that its $i-th$ component is given by (3.26):

$$(G_\sigma(x))_i = \int_{\mathbb{R}^{n-1}}\varphi(\bar{s}_i)\left(\frac{1}{\sigma}\int_{-\infty}^{+\infty} f(x_i+\sigma s_i,\bar{x}_i+\sigma\bar{s}_i)s_i\varphi(s_i)\mathrm{d}s_i\right)d\bar{s}_i$$

where (3.25)

$$g_\sigma(x_i,\bar{x}_i+\sigma\bar{s}_i) := \frac{1}{\sigma}\int_{-\infty}^{+\infty} f(x_i+\sigma s_i,\bar{x}_i+\sigma\bar{s}_i)s_i\varphi(s_i)\mathrm{d}s_i \tag{4.1}$$

is the estimate of the directional derivative of $f(x)$ along the i-th coordinate x_i and computed at the point $(x_i,\bar{x}_i+\sigma\bar{s}_i)$.

We showed that the computation of each generic entry of $G_\sigma(x)$ would be expensive, since it would require calculating the average of function (4.1) weighted by a $(n-1)$-dimensional Gaussian kernel over the subspace $\mathbb{R}^{n-1}$ of $\mathbb{R}^n$. We also discussed how Gaussian Smoothing schemes overcome this problem by rewriting $G_\sigma(x)$ as an ensemble average, and by computing its sample average approximation either using formula (3.32) or (3.33).

Instead of considering $G_\sigma(x)$ as an ensemble average, the proposed scheme starts from the following gradient estimate:

$$\overline{G}_\sigma(x) := \Big[g_\sigma(x_1,\bar{x}_1),\dots,\,g_\sigma(x_i,\bar{x}_i),\dots,g_\sigma(x_n,\bar{x}_n)\Big]^{\mathrm{T}}$$

where

$$g_\sigma(x_i,\bar{x}_i) = \frac{1}{\sigma}\int_{-\infty}^{+\infty} f(x_i+\sigma s_i,\bar{x}_i)\,s_i\,\varphi(s_i)\,\mathrm{d}s_i$$

is obtained from (4.1) with $\bar{s}_i = 0$, $i = 1,\dots,d$. This is a different result from $(G_\sigma(x))_i$ and appears to be more practical since only line integrals are involved in the formula.

Despite drastically reducing the complexity if compared to the original $G_\sigma(x)$, this estimate is still a good approximation of the gradient.

The following theorem shows that estimate $\overline{G}_\sigma(x)$ is close to $G_\sigma(x)$ and converges to it as σ tends to zero.

Theorem 4.1.1. *Let $\nabla f(x)$ be Lipschitz continuous with constant L for all $x \in \mathbb{R}^n$. Then we have that*

$$||G_\sigma(x) - \overline{G}_\sigma(x)|| \leq L\,\sigma\,\sqrt{n(15+7(n-1))}.$$

Proof: See Appendix B for the proof.

Next theorem shows that $\overline{G}_\sigma(x)$ is indeed a good approximation of the true gradient $\nabla f(x)$ and converges to it as σ tends to zero.

Theorem 4.1.2. *Let $f(x)$ be continuously differentiable for all $x \in \mathbb{R}^n$. The following holds:*

$$\lim_{\sigma\to 0} \overline{G}_\sigma(x) = \nabla f(x) \tag{4.2}$$

Proof. We prove (4.2) component-wise. By integration by parts we have

$$\begin{aligned} g_\sigma(x_i,\bar{x}_i) =& \frac{1}{\sigma}\int_{-\infty}^{+\infty} f(x_i+\sigma s_i,\bar{x}_i)\, s_i\, \varphi(s_i)\,\mathrm{d}s_i \\ =& \frac{1}{\sigma}\int_{-\infty}^{+\infty} \frac{\partial f(z_i,\bar{x}_i)}{\partial z_i}\,\frac{nz_i}{\mathrm{d}s_i}\,\varphi(s_i)\,\mathrm{d}s_i \\ =& \int_{-\infty}^{+\infty} \frac{\partial f(z_i,\bar{x}_i)}{\partial z_i}\,\varphi(s_i)\,\mathrm{d}s_i, \end{aligned} \tag{4.3}$$

where $z_i = x_i + \sigma s_i$. By changing of variable, $s_i = \frac{z_i - x_i}{\sigma}$ we obtain that

$$g_\sigma(x_i,\bar{x}_i) = \int_{-\infty}^{+\infty} \frac{\partial f(z_i,\bar{x}_i)}{\partial z_i}\,\frac{1}{\sigma}\varphi(\frac{z_i-x_i}{\sigma})\,\mathrm{d}z_i \tag{4.4}$$

and therefore, taking into account that a series of Gaussians $\frac{1}{\sigma_n}\varphi(\frac{z_i-x_i}{\sigma_n})$ with $\sigma_n \to 0$ defines a δ-dirac distribution centered in x_i [50], we have that

$$\lim_{\sigma\to 0} g_\sigma(x_i,\bar{x}_i) = \frac{\partial f(x)}{\partial x_i}.$$

□

Any entry of (4.1) is a weak definition of the derivative of $f(x)$ along x_i [50]. Note that (4.3) is well defined even though $f(x)$ is not differentiable at $(x_i,\, \bar{x}_i)$[1].

The last two theorems supports the choice of $\overline{G}_\sigma(x)$ as the starting point for an approximation scheme. Still, it is clear that it can't represent an actual way to

[1] Any L_1 function satisfying (4.3), in place of $\frac{\partial f(z_i,\bar{x}_i)}{\partial z_i}$, is a *weak derivative* of $f(x)$ along x_i.

approximate the gradient of a function. In fact, although the computation of a line integral is clearly cheaper than an integration over $\mathbb{R}^n$, it is still significantly more expensive than the computation cost of the other existing methods described in Chapter 3. In the following subsection we will show the line of reasoning and the steps that bring us from $\overline{G}_\sigma(x)$ to the actual new proposed method for computing a gradient approximation.

4.2 The NMXFD approximation scheme

We consider the functional $g_\sigma(x_i, \bar{x}_i)$ which is the $i-th$ component of the gradient estimate (4.1) and, for the sake of simplicity, we write in a single formula the result of (4.3) and (4.4).

$$\begin{aligned} g_\sigma(x_i, \bar{x}_i) =& \frac{1}{\sigma} \int_{-\infty}^{+\infty} f(x_i + \sigma s_i, \bar{x}_i)\, s_i\, \varphi(s_i)\, \mathrm{d}s_i \\ =& \int_{-\infty}^{+\infty} \frac{\partial f(z_i, \bar{x}_i)}{\partial z_i}\, \frac{1}{\sigma} \varphi(\frac{z_i - x_i}{\sigma})\, \mathrm{d}z_i \end{aligned} \tag{4.5}$$

Note that $\frac{1}{\sigma}\varphi(\frac{z_i - x_i}{\sigma})$ is $\mathcal{N}(x_i, \sigma^2)$. Our goal consists in finding a numerical approximation of the first integral in (4.5). To do that, we compute the integral in a finite range, namely between -S and S

$$\begin{aligned} \tilde{g}_\sigma(x_i, \bar{x}_i) :=& \frac{1}{\sigma} \int_{-S}^{+S} f(x_i + \sigma s_i, \bar{x}_i)\, s_i\, \varphi(s_i)\, \mathrm{d}s_i \\ =& -\frac{1}{\sigma} \int_{-S}^{+S} f(x_i + \sigma s_i, \bar{x}_i)\, \varphi'(s_i)\, \mathrm{d}s_i. \end{aligned} \tag{4.6}$$

For S sufficiently big the error between (4.5) and (4.6) is negligible due to the fast decreasing of the Gaussian to infinity, as visible from Fig. 4.1, that shows the behaviour of $\varphi'(s)$ in the range $[-5, 5]$, and Table 4.1, that shows the actual values assumed by $\varphi'(s)$ for some discrete values of s.

Table 4.1. Values assumed by $\varphi'(s)$

s	-6	-5	-4	-3	-2	-1	0
$\varphi'(s)$	3.65E-08	7.43E-06	0.000535	0.013296	0.107982	0.241971	0

The definite integral in (4.6) can be approximated by any quadrature formula, e.g. Trapezoidal Rule [9]. Dividing the interval $[-S, S]$ in $2m$ sub-intervals, each of

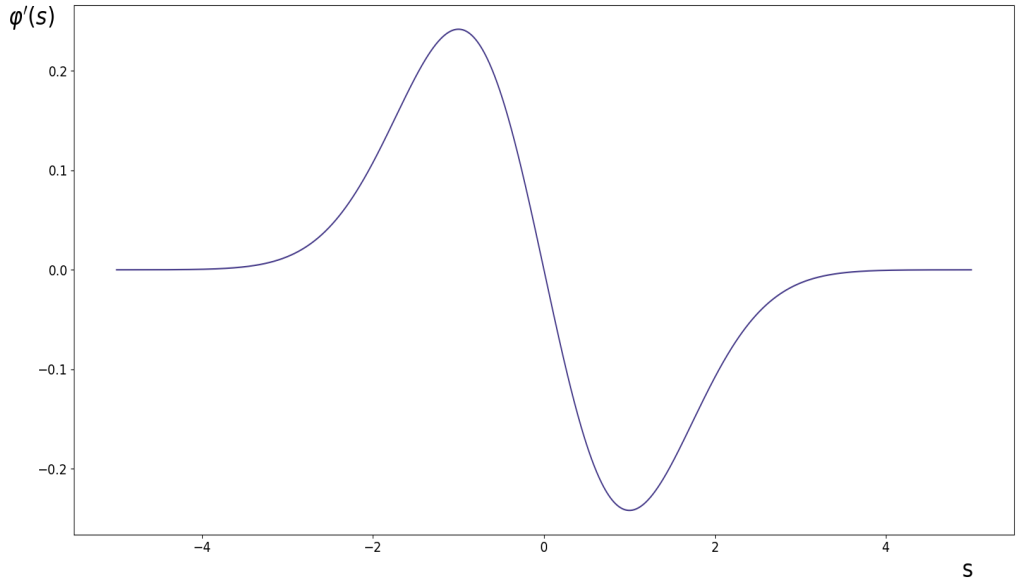

Figure 4.1. Behaviour of the Gaussian derivative

size $h = \frac{S}{m}$ we obtain:

$$\begin{aligned}\tilde{g}_\sigma(x_i, \bar{x}_i) =& \frac{h}{2\sigma}\Bigg[\Big(f(x_i - \sigma S, \bar{x}_i)\,\varphi'(-S) + f(x_i + \sigma S, \bar{x}_i)\,\varphi'(S) + \\ & 2\sum_{j=1}^{2m-1} f(x_i + \sigma(-S + jh), \bar{x}_i)\,\varphi'(-S + jh)\Big)\Bigg] + \\ & + \frac{h^2}{6\sigma}\frac{S}{d\tau^2} f(x_i + \sigma\tau, \bar{x}_i)\,\varphi'(\tau) \qquad \tau \in [-S, S]. \end{aligned} \tag{4.7}$$

It is well known that, under very general conditions, the trapezoidal quadrature formula (4.7) has an error that is $\mathcal{O}(1/m^2)$ [23]. Indeed, once σ and S are chosen, we can easily check this property in our case. Let

$$\begin{aligned}\epsilon_\sigma(\tau, m) &= \frac{h^2}{6\,\sigma}\frac{S}{d\tau^2} f(x_i + \sigma\tau, \bar{x}_i)\,\varphi'(\tau) \qquad \tau \in [-S, S]. \\ &= \frac{S^3}{6\,\sigma\,m^2}\frac{d}{d\tau^2} f(x_i + \sigma\tau, \bar{x}_i)\,\varphi'(\tau) \qquad \tau \in [-S, S].\end{aligned}$$

Note that the derivatives of a guassian kernel $|\varphi^{(k)}(\tau)|$, up to the third order, are all less than 1 in absolute value for any τ, and decrease rapidly as τ increases. Therefore, for f sufficiently smooth in $(x_i \pm \sigma\, S)$, let

$$K(x_i) = max\left(|f(x_i + \sigma\tau, \bar{x}_i)|, |\frac{\mathrm{d}}{\mathrm{d}\tau} f(x_i + \sigma\tau, \bar{x}_i)|, |\frac{\mathrm{d}^2}{\mathrm{d}\tau^2} f(x_i + \sigma\tau, \bar{x}_i)|\right).$$

We can write:

$$|\epsilon_\sigma(\tau, m)| \leq \frac{h^2\, S}{6\sigma} K(x_i) = \frac{S^3}{6\sigma\, m^2} K(x_i), \quad \tau \in [-S, +S]$$

Let us rewrite (4.7) as follows

$$\tilde{g}_\sigma(x_i, \bar{x}_i) = \bar{g}_\sigma(x_i, \bar{x}_i) + \epsilon_\sigma(\tau, m)$$

The larger the number of function evaluation m, the smaller the error term $\epsilon_\sigma(\tau, m)$. On the other hand, $\bar{g}_\sigma(x_i)$ can be interpreted as a combination of finite differences with some coefficients. Keeping in mind that $\varphi'(t) = -\varphi'(-t)$ and that $\varphi'(0) = 0$, after some simple algebra we can write:

$$\begin{aligned}\bar{g}_\sigma(x_i, \bar{x}_i) = -\frac{h}{2\sigma}\Bigg[&|\varphi'(m\,h)|\Big(f(x_i - \sigma\, m\, h, \bar{x}_i) - f(x_i + \sigma\, m\, h, \bar{x}_i)\Big) \\ &+2\sum_{j=1}^{m-1} |\varphi'(jh)|\Big(f(x_i - \sigma jh, \bar{x}_i) - f(x_i + \sigma jh, \bar{x}_i)\Big)\Bigg]\end{aligned}$$

from which

$$\begin{aligned}\bar{g}_\sigma(x_i, \bar{x}_i) = \frac{h}{2\sigma}\Bigg[&|\varphi'(m\,h)|\, 2\sigma\, m\, h\, \frac{f(x_i + \sigma\, m\, h, \bar{x}_i) - f(x_i - \sigma\, m\, h, \bar{x}_i)}{2\sigma\, m\, h} \\ &+2\sum_{j=1}^{m-1} |\varphi'(jh)|\, 2\sigma\, j\, h\, \frac{f(x_i + \sigma jh, \bar{x}_i) - f(x_i - \sigma jh, \bar{x}_i)}{2\sigma\, j\, h}\Bigg].\end{aligned} \tag{4.8}$$

It is clear that $\bar{g}_\sigma(x_i, \bar{x}_i)$ is a linear combination of finite difference approximations, with different step sizes; for $\sigma h \to 0$, each one converges to the true value of the partial derivative $\partial f(x_i, \bar{x}_i)/\partial x_i$. Therefore, the estimate $\bar{g}_\sigma(x_i, \bar{x}_i)$ converges to the true value only if the sum of its coefficients equals one. For this reason, it is advisable to *normalize* the coefficients of the linear combination in (4.8) to eliminate the estimate bias for σ finite.
To this aim, let C be the sum of all the coefficients:

$$\left.\begin{aligned} &C = \textstyle\sum_{j=1}^{m} a'_j, \\ &a'_j = 2\, j\, h^2\, |\varphi'(jh)|, \quad j = 1, \ldots, m-1, \\ &a'_m = m\, h^2\, |\varphi'(mh)| \end{aligned}\right\} \tag{4.9}$$

We can then write the normalized version of (4.8) as:

$$\bar{g}_\sigma(x_i, \bar{x}_i) = \sum_{j=1}^{m} a_j\, \frac{f(x_i + \sigma\, j\, h, \bar{x}_i) - f(x_i - \sigma\, j\, h, \bar{x}_i)}{2\sigma\, j\, h} \tag{4.10}$$

where

$$a_j = \frac{a'_j}{C}, \qquad \sum_{j=1}^{m} a_j = 1. \tag{4.11}$$

For σ small enough the normalization of the coefficients may not be necessary, the distorsion of the estimate being negligible. Let us now evaluate the error bound corresponding to estimate (4.10), from here on referred to as NMXFD (Normalized Mixed Finite Difference).

Theorem 4.2.1. *Let $f(x)$ be twice continuously differentiable and its Hessian be H-Lipschitz for all $x \in \mathbb{R}^n$. Consider the gradient approximation obtained by (4.10)*

$$\widehat{G}_\sigma(x) = [\, \hat{g}_\sigma(x_1), \ldots \hat{g}_\sigma(x_n)\,]^{\mathrm{T}} \tag{4.12}$$

We have that

$$\|\widehat{G}_\sigma(x) - \nabla f(x)\| \leq \sqrt{n}\, \frac{H\sigma^2\, S^2}{6} \tag{4.13}$$

Proof. Any single finite difference term in (4.10) has an error with respect to the true value $\partial f(x_i, \bar{x}_i)/\partial x_i$ whose bound depends on the step size and on the regularity properties of function f. From [17], we have that

$$\left| \frac{f(x_i + \sigma\, j\, h, \bar{x}_i) - f(x_i - \sigma\, j\, h, \bar{x}_i)}{2\sigma\, j\, h} - \frac{\partial f(x_i, \bar{x}_i)}{\partial x_i} \right| \leq \frac{H\sigma^2 (jh)^2}{6}$$

for $j = 1, \ldots, m$. Therefore, since $\sum_{j=1}^{m} a_j = 1$, and $a_j > 0$, $j = 1, \ldots, m$, we can write

$$\begin{aligned}
\left| \hat{g}_\sigma(x_i) - \frac{\partial f(x_i, \bar{x}_i)}{\partial x_i} \right| &= \left| \hat{g}_\sigma(x_i) - \sum_{j=1}^{m} a_j\, \frac{\partial f(x_i, \bar{x}_i)}{\partial x_i} \right| \\
&\leq \sum_{j=1}^{m} a_j \left| \frac{f(x_i + \sigma\, j\, h, \bar{x}_i) - f(x_i - \sigma\, j\, h, \bar{x}_i)}{2\sigma\, j\, h} - \frac{\partial f(x_i, \bar{x}_i)}{\partial x_i} \right| \\
&\leq \frac{H\sigma^2\, h^2}{6} \left(\sum_{j=1}^{m} a_j\, j^2 \right) \leq \frac{H\sigma^2\, h^2\, m^2}{6} = \frac{H\sigma^2\, S^2}{6}.
\end{aligned}$$

which, applied to all entries of $\widehat{G}_\sigma(x) - \nabla f(x)$, proves the theorem. □

Here we used the equality $m\, h = S$ that implies that the error bound does not depend on the number of function evaluations.

4.2.1 NMXFD with noisy data

Let us now evaluate how the performance of the gradient estimate *NMXFD* (4.12) – here referred to as $\hat{G}_\sigma^{\mathrm{MXF}}(x)$ – compares with that of the *CFD*, taking also into account the presence of an additive noise affecting the sampled function values $f(x)$.

Let $\{e_i\}$ be the canonical basis of $\mathbb{R}^n$, then we can write:

$$\hat{G}_\sigma^{\mathrm{MXF}}(x) = \sum_{i=1}^{n} \hat{g}_\sigma(x_i)\, e_i. \tag{4.14}$$

With the same notation, denoting with σh the step-size, we can easily write the gradient estimate according to the *CFD* scheme, here denoted as $\hat{G}_\sigma^{\mathrm{CFD}}(x)$:

$$\hat{G}_\sigma^{\mathrm{CFD}}(x) = \sum_{i=1}^{n} \frac{f(x_i + \sigma\, h, \bar{x}_i) - f(x_i - \sigma\, h, \bar{x}_i)}{2\sigma\, h}\, e_i \tag{4.15}$$

We assume that the sampled function values are affected by additive noise, as modelled in (3.1).
We now compute the estimation errors for the two schemes and compare them in terms of accuracy (mean value) and precision (variance).

The NMXFD scheme
In the presence of noise, according to (4.14), a number $N = 2mn$ of function evaluations is considered to obtain

$$\hat{G}_\sigma^{\mathrm{MXF}}(x) = \sum_{i=1}^{n}\sum_{i=1}^{n} \hat{g}_\sigma(x_i)\, e_i + \sum_{i=1}^{n} \left(\sum_{j=1}^{m} a_j\, \frac{\epsilon_{i,j}^{+} - \epsilon_{i,j}^{-}}{2\sigma\, j\, h} \right) e_i$$

with $\epsilon_{i,j}^{\pm}$ denoting the error terms on the function values $\phi(x_i \pm \sigma\, jh, \bar{x}_i)$, $i = 1, \ldots, n$, $j = 1, \ldots, m$. For the estimation error

$$e_{\mathrm{MXF}}(x) = \hat{G}_\sigma^{\mathrm{MXF}}(x) - \nabla f(x),$$

we readily obtain that

$$\begin{aligned} E\left[e_{\mathrm{MXF}}(x)\right] &= \sum_{i=1}^{n}\sum_{i=1}^{n} \hat{g}_\sigma(x_i)\, e_i - \nabla f(x), \\ var\left[e_{\mathrm{MXF}}(x)\right] &= \frac{n\,\lambda^2}{2\sigma^2 h^2} \left(\sum_{j=1}^{m} \frac{a_j^2}{j^2} \right). \end{aligned} \tag{4.16}$$

Under the assumptions of Thm. (4.2.1), and taking into account (4.13), we obtain

$$\| E\left[e_{\mathrm{MXF}}(x)\right] \| \leq \sqrt{n}\, \frac{H\sigma^2\, m^2\, h^2}{6}. \tag{4.17}$$

The CFD scheme

In the presence of noise, according to (4.15), a number $N = 2n$ of function evaluations is needed to obtain the gradient estimate:

$$\hat{G}_\sigma^{\mathrm{CFD}}(x) = \sum_{i=1}^{n} \frac{\phi(x_i + \sigma\, h, \bar{x}_i) - \phi(x_i - \sigma\, h, \bar{x}_i)}{2\sigma\, h}\, e_i + \sum_{i=1}^{n} \frac{\epsilon_i^+ - \epsilon_i^-}{2\sigma\, h}\, e_i$$

with $\epsilon_i^\pm$ denoting the error terms on the function values $\phi(x_i \pm \sigma\, h, \bar{x}_i)$, $i = 1, \ldots, n$.

Let

$$e_{\mathrm{CFD}}(x) = \hat{G}_\sigma^{\mathrm{CFD}}(x) - \nabla f(x)$$

be the estimation error. Recalling the results shown in subsection 3.1.1, we have that, for functions f that are twice continuously differentiable with a H-Lipschitz continuous Hessian

$$\begin{aligned} E[e_{\mathrm{CFD}}(x)] &= \sum_{i=1}^{n} \frac{\phi(x_i + \sigma\, h, \bar{x}_i) - \phi(x_i - \sigma\, h, \bar{x}_i)}{2\sigma\, h}\, e_i, \\ var[e_{\mathrm{CFD}}(x)] &= n\, \frac{2\lambda^2}{4\,\sigma^2\, h^2} = \frac{n\,\lambda^2}{2\sigma^2\, h^2}, \\ \|E[e_{\mathrm{CFD}}(x)]\| &\leq \sqrt{n}\, \frac{H\,\sigma^2\, h^2}{6}. \end{aligned}$$

As for the error variance, two interesting results can be proved.

Proposition 4.2.2. *For any $m > 1$, the variance of the estimation error of the NMXFD scheme is strictly lower than the variance of the estimation error of the CFD scheme, i.e,.*

$$var\,[e_{MXF}(x)] < var\,[e_{CFD}(x)]$$

in any $x \in \mathbb{R}^n$ and for any σ, h.

Proof. The sum of squares $\sum_{j=1}^{m} a_j^2$ is strictly less then 1 since the coefficients a_j, $j = 1, \ldots, m$, are all positive and their sum is 1. Therefore from (4.16) we obtain that

$$var\,[e_{\mathrm{MXF}}(x)] = \frac{n\lambda^2}{2\sigma^2\, h^2} \sum_{j=1}^{m} \frac{a_j^2}{j^2} < \frac{n\lambda^2}{2\sigma^2\, h^2} = var\,[e_{\mathrm{CFD}}(x)]\,.$$

□

Now we further show that $var\,[e_{\mathrm{MXF}}(x)]$ goes to zero as N increases.

Proposition 4.2.3. *For any $x \in \mathbb{R}^n$, the variance of the estimation error of the NMXFD scheme has the following asymptotic behavior*

$$var\,[e_{MXF}(x)] \sim \mathcal{O}\left(\frac{1}{N}\right). \tag{4.18}$$

Proof. By taking into account relations (4.9), we have that

$$\begin{aligned} C =& m\,h^2 \left(|\varphi'(mh)| + 2\sum_{j=1}^{m-1} \frac{j}{m} |\varphi'(jh)| \right) \\ &\leq 2m\,h\,\frac{h}{2} \left(|\varphi'(mh)| + 2\sum_{j=1}^{m-1} |\varphi'(jh)| \right). \end{aligned} \tag{4.19}$$

Let us denote with $I^{(1)}_{\varphi'}(m)$ the following quantity

$$I^{(1)}_{\varphi'}(m) = \frac{h}{2} \left(|\varphi'(mh)| + 2\sum_{j=1}^{m-1} |\varphi'(jh)| \right)$$

that is the trapezoidal quadrature formula for the integral

$$\int_0^S |\varphi'(t)|\,dt = \frac{1}{\sqrt{2\pi}} \left(1 - e^{-\frac{S^2}{2}} \right).$$

Due to the $\mathcal{O}(1/N^2)$ property of the error of the trapezoidal rule, we have that

$$\left| I^{(1)}_{\varphi'}(m) - \frac{1}{\sqrt{2\pi}} \left(1 - e^{-\frac{S^2}{2}} \right) \right| = \mathcal{O}(1/N^2).$$

Therefore, from (4.19), we easily obtain that

$$\left| C - \frac{2m\,h}{\sqrt{2\pi}} \left(1 - e^{-\frac{S^2}{2}} \right) \right| \leq 2m\,h \left| I^{(1)}_{\varphi'}(m) - \frac{1}{\sqrt{2\pi}} \left(1 - e^{-\frac{S^2}{2}} \right) \right| = \mathcal{O}(1/N^2) \tag{4.20}$$

so that C is a bounded quantity as $N = 2m\,n$ increases (by increasing m), taking into account that $mh = S$. Now, according to the relations (4.11) we can write

$$\begin{aligned} \sum_{j=1}^{m} \frac{a_j^2}{j^2} &= \frac{1}{C^2} \left(\frac{m^2\,h^4 |\varphi'(mh)|^2}{m^2} + \sum_{j=1}^{m-1} \frac{4\,j^2\,h^4\,|\varphi'(jh)|^2}{j^2} \right) \\ &= \frac{h^4}{C^2} \left(|\varphi'(mh)|^2 + 2\sum_{j=1}^{m-1} 2|\varphi'(jh)|^2 \right) \\ &\leq \frac{2\,h^3}{C^2} \frac{h}{2} \left(2|\varphi'(mh)|^2 + 2\sum_{j=1}^{m-1} 2|\varphi'(jh)|^2 \right). \end{aligned}$$

Define now $I^{(2)}_{\varphi'}(m)$ as follows

$$I^{(2)}_{\varphi'}(m) = \frac{h}{2} \left(2|\varphi'(mh)|^2 + 2\sum_{j=1}^{m-1} 2|\varphi'(jh)|^2 \right).$$

It is the trapezoidal quadrature rule for the integral

$$2\int_0^S |\varphi'(t)|^2\,dt = \sqrt{\pi}\,\mathrm{erf}(S) - Se^{-S^2} = \Phi(S),$$

where $\mathrm{erf}(z) = \frac{2}{\sqrt{\pi}} \int_0^z e^{-t^2}\, dt$ is the Gauss error function. Hence, for the usual property of the error, we can write

$$\left| I^{(2)}_{\varphi'}(m) - \Phi(S) \right| = \mathcal{O}(1/N^2).$$

Therefore, we obtain that

$$\begin{aligned} var\left[e_{\mathrm{MXF}}(x)\right] =& \frac{n\,\lambda^2}{2\sigma^2\,h^2} \left(\sum_{j=1}^{m} \frac{a_j^2}{j^2} \right) \leq \frac{n\,\lambda^2}{2\sigma^2\,h^2} \frac{2\,h^3}{C^2} I^{(2)}_{\varphi'}(m) \\ \leq& \frac{n\,\lambda^2}{\sigma^2} \frac{h}{C^2} \left(\left| I^{(2)}_{\varphi'}(m) - \Phi(S) \right| + \left| \Phi(S) \right| \right). \end{aligned}$$

Now recalling that $m\,h = S$, and that $N = 2m\,n$, we can write

$$\begin{aligned} var\left[e_{\mathrm{MXF}}(x)\right] \leq& \frac{n\,\lambda^2}{\sigma^2} \frac{S}{m\,C^2} \left(\left| I^{(2)}_{\varphi'}(m) - \Phi(S) \right| + \left| \Phi(S) \right| \right) \\ \leq& \frac{2}{N} \frac{n^2\,\lambda^2\,S}{\sigma^2\,C^2} \left(\left| I^{(2)}_{\varphi'}(m) - \Phi(S) \right| + \left| \Phi(S) \right| \right) \end{aligned}$$

which, along with (4.20), proves the proposition. □

This result suggests that performing a combination of finite differences in the noisy setting yields to improvement in the quality of the gradient estimation. The reader might wonder how the variance of *NMXFD* compares with the one of a method that uses the same budget of function evaluations, and not just $2n$ like the *CFD* scheme. In the following subsection, we will show this comparison with a very simple method: the average of several *CFDs*.

Comparison with multiple CFDs

The simplest combination of finite differences possible is the average of a number m of multiple *CFDs* ($mCFD$) computed over repeated measures

$$\hat{G}^{\mathrm{mCFD}}_{\sigma}(x) = \frac{1}{m} \sum_{k=1}^{m} \hat{G}^{\mathrm{CFD}}_{\sigma,k}(x) \tag{4.21}$$

where $\hat{G}^{\mathrm{CFD}}_{\sigma,k}(x)$ is the *CFD* in (4.15) computed at the same points, but with a different independent realization k of the noise. This formula, obviously, reduces the error variance of *CFD* by $1/m$. It then becomes interesting to see if

$$var\left[e_{\mathrm{MXF}}(x)\right] = \frac{n\lambda^2}{2\sigma^2\,h^2} \sum_{j=1}^{m} \frac{a_j^2}{j^2} < \frac{1}{m} \frac{n\lambda^2}{2\sigma^2\,h^2} = var\left[e_{\mathrm{mCFD}}(x)\right]. \tag{4.22}$$

Because of the complicated structure of the coefficients a_j a formal proof of (4.22) can be involved. In Fig. 4.2 we show a visual verification of (4.22) for increasing

values of m, with a uniform sampling within the range [-S, S] with $S = mh = 3$ to compute coefficients a_j. In Table 4.2 we also report the numerical values associated to each value of m. For each value of m, we highlight in bold the smaller coefficient, the one that yields to a greater reduction of the error variance.

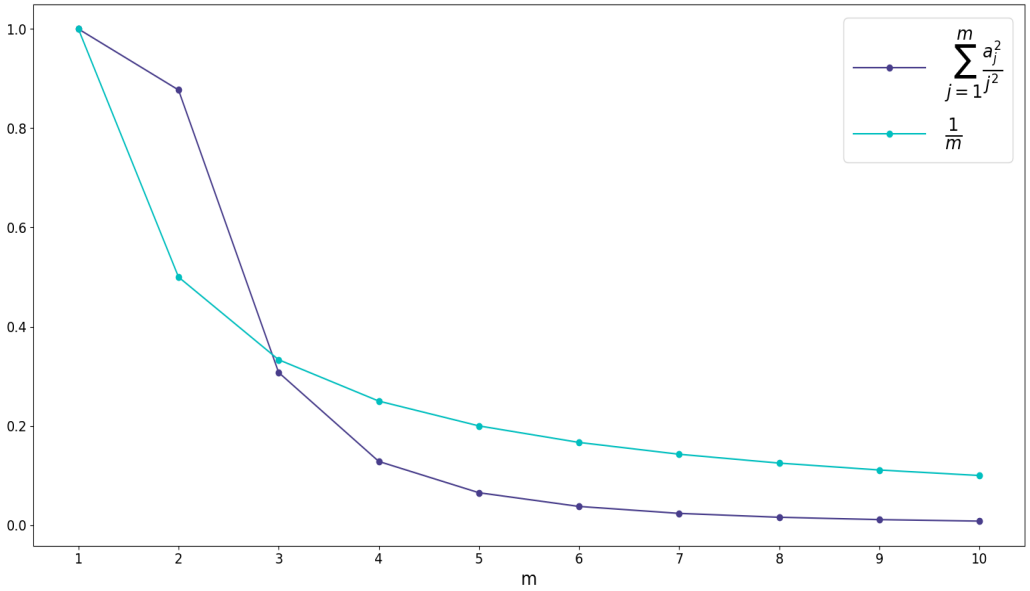

Figure 4.2. Coefficients effect on variance reduction: NMXFD vs mCFD, S = 3

For $m = 1$ the reduction of the variance of the two methods is the same. For all $m > 2$, we can see that the reduction of the error variance of *NMXFD* is greater than that of *mCFD*.

Similar results can be obtained for different values of the parameter S. Fig. 4.3 shows the visual verification of (4.22) for increasing values of m, when we set $S = mh = 2$ as the value for the range [-S, S] used to compute coefficients a_j.

We also report the associated Table 4.3 with the numerical values associated to each value of m. The table has the same structure of Table 4.2.

In the second scenario reported, when $S = 2$, we can see that for $m = 2$ the variance reduction obtained using the NMXFD method is almost the same as the one of the mCFD method. This is different from what happens in the case of $S = 3$, where the variance reduction obtained with the mCFD is significantly smaller that that of NMXFD.

This make sense from an intuitive point of view, if we focus on the role of the parameter S in the computation of the gradient estimate using the proposed NMXFD approach. Using a smaller value of S means truncating the integral (4.5) in a smaller range. Therefore, fewer points are needed to obtain an accurate approximation of the integral value.

Table 4.2. Variance reduction coefficient on increasing m for NMXFD (1^{st} column) and mCFD (2^{nd} column), when S = mh = 3.

m	$\sum_{j=1}^{m} \frac{a_j^2}{j^2}$	$\frac{1}{m}$
1	1	1
2	0.877023	**0.5**
3	**0.307637**	0.333333
4	**0.128374**	0.25
5	**0.065331**	0.2
6	**0.037682**	0.166667
7	**0.023683**	0.142857
8	**0.015845**	0.125
9	**0.011119**	0.111111
10	**0.008101**	0.1

Table 4.3. Variance reduction coefficient on increasing m for NMXFD (1^{st} column) and mCFD (2^{nd} column), when S = mh = 2.

m	$\sum_{j=1}^{m} \frac{a_j^2}{j^2}$	$\frac{1}{m}$
1	1	1
2	0.501889	**0.5**
3	**0.145629**	0.333333
4	**0.061182**	0.25
5	**0.031303**	0.2
6	**0.018119**	0.166667
7	**0.011415**	0.142857
8	**0.007651**	0.125
9	**0.005376**	0.111111
10	**0.003921**	0.1

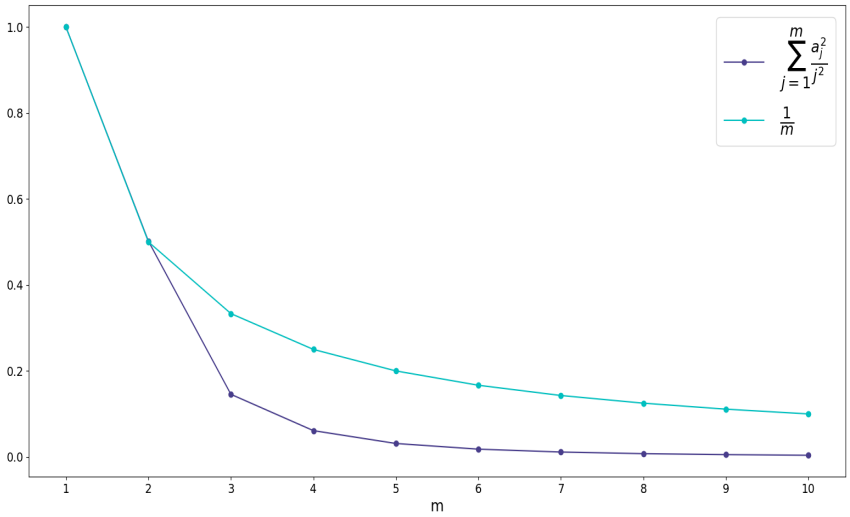

Figure 4.3. Coefficients effect on variance reduction: NMXFD vs mCFD, S = 2

4.3 Numerical experiments – point-wise comparison

We tested our method for estimating the gradient by comparing its performance with those of other methods on 69 functions from the Schittkowski test set [120]. For each function, we did the following: we generated a random starting point x^0 and minimized the function using the quasi-Newton method of Broyden, Fletcher, Goldfarb and Shanno (BFGS) [100], finding the optimal point x^* with $\nabla f(x^*) \approx 0$. We then identified the first instance of a point x^k where

$$\frac{\|\nabla f(x^k)\|}{\|\nabla f(x^0)\|} \leq \alpha$$

for each of the following values of α: $10^0, 10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}$. In this way, we generated seven different buckets, one for each α, of 69 different points, one for each function. Bucket i indicates the one associated to $\alpha = 10^{-i}$. Bucket 0 is therefore the one with the points that are farther from the optimal solution and bucket 6 is the one with points closer to the optimal solution.
Then, for each point we computed the gradient approximations obtained with the Normalized MiXed Finite Differences scheme (*NMXFD*) and with those considered benchmarks by the literature, namely: Forward Finite Differences (*FFD*), Central Finite Differences (*CFD*), Gaussian Smoothed Gradient (*GSG*), Central Gaussian Gmoothed Gradient (*cGSG*) as defined in Sect. 3.1.2. Different tables will summarize the results of this comparison.

The tables show, for different values of the number of function evaluations (N) and different buckets (B), the median value of the log of the relative approximation

error over all the 69 points in each bucket.
We define relative approximation error as

$$\eta = \frac{\|g(x) - \nabla f(x)\|}{\|\nabla f(x)\|}$$

where $g(x)$ is the generic gradient estimate.

The number of function evaluations N is expressed in the following tables as a function of the number of dimensions n. *FFD* and *CFD* schemes only allow for a specific value of N ($n+1$ and $2n$, respectively). In *GSG* and in *cGSG*, N is linked to the number of direction sampled to build the gradient approximation ($N = (M+1)$ in (3.32) and $N = 2\,M$ in (3.33)). In the *NMFXD* scheme, the value of N is linked to the value of m in formula (4.10). In particular, we have that $N = 2mn$. In each table, the lowest entry for every bucket is highlighted in bold, and the second lowest is italicized. Finally, we will show the results obtained both in the noise-free scenario and in the noisy one.

4.3.1 Noise free setting

For the noise-free setting we report three different tables obtained using a different value of σ (shared by all the schemes) to compute the gradient approximation.

Table 4.4. Median log of relative error with $\sigma = 10^{-2}$

Scheme	N	B0	B1	B2	B3	B4	B5	B6
FFD	n+1	0.08	1.22	2.20	3.43	4.43	5.47	6.52
CFD	2n	**-2.26**	**-1.13**	**-0.32**	**0.69**	**1.60**	**2.41**	**3.58**
	2n+1	1.84	1.92	2.52	3.57	4.69	5.63	6.92
GSG	4n+1	1.70	1.78	2.34	3.41	4.50	5.46	6.82
	8n+1	1.54	1.66	2.10	3.31	4.49	5.38	6.67
	2n	1.97	1.96	1.96	1.99	2.08	2.44	3.46
cGSG	4n	1.86	1.82	1.86	1.90	2.06	2.95	4.01
	8n	1.71	1.69	1.74	1.81	2.13	3.14	4.28
	2n	-1.66	-0.53	0.28	1.29	2.26	3.06	4.18
NMXFD	4n	-1.98	-0.84	-0.03	0.97	1.92	2.71	3.87
	8n	*-1.99*	*-0.86*	*-0.05*	*0.96*	*1.90*	*2.68*	*3.85*

Table 4.5. Median log of relative error with $\sigma = 10^{-5}$

Scheme	N	B0	B1	B2	B3	B4	B5	B6
FFD	n+1	-2.92	-1.78	-0.80	0.43	1.43	2.47	3.51
CFD	2n	**-8.17**	**-6.99**	**-6.14**	**-4.87**	**-3.75**	**-3.07**	**-1.73**
GSG	2n+1	1.84	1.84	1.85	1.84	2.00	2.64	3.92
	4n+1	1.69	1.71	1.71	1.74	1.90	2.48	3.80
	8n+1	1.53	1.57	1.56	1.57	1.77	2.41	3.67
cGSG	2n	1.96	1.96	1.96	1.96	1.96	1.97	1.93
	4n	1.86	1.82	1.85	1.85	1.85	1.85	1.83
	8n	1.71	1.68	1.70	1.68	1.71	1.71	1.71
NMXFD	2n	-7.66	-6.47	-5.58	-4.43	-3.24	-2.75	-1.21
	4n	-7.90	-6.74	-5.85	-4.67	-3.55	-2.79	-1.45
	8n	*-7.95*	*-6.76*	*-5.87*	*-4.73*	*-3.57*	*-2.84*	*-1.54*

Table 4.6. Median log of relative error with $\sigma = 10^{-8}$

Scheme	N	B0	B1	B2	B3	B4	B5	B6
FFD	n+1	-5.56	-4.73	-3.74	1.43	-1.50	-0.44	0.67
CFD	2n	-6.00	-6.20	-6.23	**-3.75**	-6.20	-6.25	**-6.23**
GSG	2n+1	1.84	1.84	1.84	2.00	1.82	1.83	1.91
	4n+1	1.69	1.71	1.72	1.90	1.70	1.69	1.79
	8n+1	1.53	1.57	1.56	1.77	1.55	1.55	1.66
cGSG	2n	1.96	1.96	1.96	1.96	1.96	1.96	1.93
	4n	1.86	1.82	1.85	1.85	1.84	1.85	1.82
	8n	1.71	1.68	1.70	1.71	1.71	1.71	1.71
NMXFD	2n	-6.48	-6.36	-6.52	-3.24	-6.41	-6.42	-6.09
	4n	*-6.17*	*-6.29*	*-6.41*	-3.55	*-6.43*	*-6.48*	*-6.20*
	8n	**-6.42**	**-6.44**	**-6.44**	*-3.57*	**-6.50**	**-6.51**	-6.15

It is possible to notice that in a noise-free setting, lower values of σ tend to yield to better results, as one would expect from the theory. The closer the point is to the minimum value of a function, the harder it is to obtain an accurate estimate of its gradient, unless σ is very small. As a matter of fact, for points belonging to lower index buckets–thus far from the minimum of the function, the value $\sigma = 10^{-5}$ yields the better performances, while accurate estimates of the gradient of points closer to the minimum value of a function require using of a lower value of σ. We can also see that the error of the proposed method, *NMXFD*, is of the same order of magnitude of that of *CFD*, and almost always better than that of the other methods.

In our experiments, we have also produced gradient estimates using two more methods:

- by removing the normalization of the coefficients in the computation of *NMXFD*, i.e., implementing the gradient approximation as in (4.8).
- by computing the estimate as the raw average of central finite differences at different stepsizes, that is (4.10) with $a_j = \frac{1}{m}$.

Both of these methods performed consistently worse than *NMXFD*, and they have not been reported in the tables for brevity. Still, the better performances of *NMXFD* over the raw average of central finite differences seem to confirm that the rationale behind the choice of coefficients used to weight the *CFDs* in the proposed approach is promising from a computational point of view.

4.3.2 Noisy setting

In this subsection we report the results obtained in the noisy scenario, where the noise term is described at the beginning of Chapter 3 and has $\lambda = 0.001$. The estimation procedure is slightly different from the one of the noise-free setting. In Table 4.7, the median log of the relative errors η_i of the 69 different Schittkowski function is reported. Each η_i is computed as the average of 100 relative approximation errors, resulting from 100 independent noise realizations. The rationale behind this choice was to mitigate the dependence of the results from one particular noise realization. Results are shown in Table 4.7, where the gradient estimates are obtained with $\sigma = 0.01$.

Table 4.7 shows that *NMXFD* performs better than the other schemes in presence of noise, although reasonably low relative approximation errors are obtained only for the first three buckets. For the other ones the error η increases significantly. This is due to the fact that the denominator of η gets smaller as we move to points close to the minimum value of the function, while the variance of the approximation error does not change across different buckets. Just like in the noise-free setting, increasing the number of function evaluations allows to increase the precision of all the schemes, as expected from the theory.

Table 4.7. Median log of relative error with $\sigma = 10^{-2}$, noisy setting.

Scheme	N	B0	B1	B2	B3	B4	B5	B6
FFD	n+1	0.22	1.45	2.46	3.57	4.86	5.74	6.86
CFD	2n	-1.06	0.10	1.34	2.23	3.50	4.32	5.49
GSG	4n+1	1.72	1.79	2.56	3.66	4.82	5.69	6.84
	8n+1	1.56	1.66	2.43	3.51	4.67	5.56	6.70
	12n+1	1.47	1.56	2.35	3.43	4.59	5.48	6.61
cGSG	4n	1.85	1.85	1.86	2.39	3.61	4.33	5.65
	8n	1.71	1.71	1.73	2.29	3.55	4.28	5.61
	12n	1.62	1.63	1.65	2.24	3.52	4.25	5.58
NMXFD	4n	-1.22	0.00	1.17	2.23	3.43	4.24	5.52
	8n	*-1.31*	*-0.09*	*1.08*	*2.15*	*3.40*	*4.19*	*5.42*
	12n	**-1.36**	**-0.15**	**1.05**	**2.11**	**3.39**	**4.15**	**5.38**

Different values of σ for estimating the gradient (10^{-1} , 10^{-3}, 10^{-4}) have also been used. The associated tables have not been reported for brevity, since they yielded to the same conclusions and since the performances for almost every method and every bucket with those values of σ are significantly worse. This can be inferred from the theory, since the value of σ influences the bias and the variance of the estimate error in opposite directions, as we can see from (4.16) and (4.17) in Sect. 4.2.1.

The numerical experiments show the good performances of the proposed method when compared with those of the standard methods commonly used in the literature. In particular, the performances of *NMXFD* are comparable with those of *CFD* in absence of noise and better with noisy data and are better than those of other schemes in both scenarios.

The results seem to confirm the idea that performing a combination of finite differences in the noisy setting increases the quality of the gradient estimation.

In Table 4.8 we finally report the comparison of the median log of relative error between *NMXFD* and $\hat{G}_{\sigma}^{\mathrm{mCFD}}$ (the combination of *CFDs* defined in (4.21)) on increasing noise levels λ, all computed with a value σ of 0.01 and always using the same function evaluation budget. We do not report the performances of other methods for brevity, since they confirm the same conclusions provided by Table 4.7.

Table 4.8. Median log of relative error with $\sigma = 10^{-2}$, different values of λ

	Scheme	N	B0	B1	B2	B3	B4	B5	B6
$\boldsymbol{\lambda = 0.001}$		**4n**	-1.22	-0.05	1.17	2.13	3.35	4.18	5.39
	mCFD	**8n**	-1.36	-0.19	1.02	2.05	3.2	4.07	5.32
		12n	**-1.43**	**-0.23**	**0.94**	**1.99**	**3.16**	**4.01**	**5.28**
		4n	-1.22	0	1.17	2.23	3.43	4.24	5.52
	NMXFD	**8n**	-1.31	-0.09	1.08	2.15	3.4	4.19	5.42
		12n	-1.36	-0.15	1.05	2.11	3.39	4.15	5.38
$\boldsymbol{\lambda = 0.01}$		**4n**	-0.24	0.93	2.1	3.04	4.33	5.14	6.31
	mCFD	**8n**	-0.37	0.78	2.02	2.92	4.17	4.99	6.16
		12n	-0.47	0.71	1.93	**2.82**	4.09	4.91	6.08
		4n	-0.3	0.89	2.09	3	4.27	5.09	6.25
	NMXFD	**8n**	-0.39	0.78	2.01	2.92	4.16	4.97	6.16
		12n	**-0.48**	**0.67**	**1.91**	**2.82**	**4.07**	**4.88**	**6.06**
$\boldsymbol{\lambda = 0.1}$		**4n**	0.76	1.93	3.05	4.03	5.32	6.03	7.29
	mCFD	**8n**	0.6	1.78	2.9	3.89	5.16	5.87	7.15
		12n	0.52	1.7	2.8	3.8	5.08	5.79	7.06
		4n	0.69	1.88	2.97	3.98	5.26	5.97	7.23
	NMXFD	**8n**	0.58	1.77	2.89	3.88	5.15	5.86	7.12
		12n	**0.49**	**1.66**	**2.77**	**3.79**	**5.05**	**5.77**	**7.03**

Table 4.8 shows that the basic combination $\hat{G}_\sigma^{\mathrm{mCFD}}$ is indeed a good gradient approximation due to the effect of the average that reduces the error variance. As the noise level increases, $\hat{G}_\sigma^{\mathrm{MXF}}$ tends to be better than $\hat{G}_\sigma^{\mathrm{mCFD}}$. This supports the idea that a good gradient approximation depends on both the coefficients of the linear combination and the sampling points where the differences are computed. In this respect, the analysis developed in Sect. 4.2 to define the new gradient estimate, provides a guide to design a more efficient estimate, depending on the following points:

- the parameter S that determines the range of integration in integral (4.6);
- the integration formula used to approximate integral (4.6);
- the filter parameter σ;
- the sampling strategy of the function within the integration range $(-S, S)$.

In this early investigation, we heuristically tried several values for the parameters S and σ, without trying different integration formulas or sampling criteria. The choice of σ may be difficult and affects the quality of the approximation. When the noise level is known, there are some strategies to make a proper choice of σ as in [122]. When the noise level is not known, the choice of this parameter becomes harder and represents an open question to be further investigated, along with the other points in the list above, to improve the performances of NMXFD.
In the next section, we show two experiments where we analyze the impact of the obtained gradient approximation on optimization algorithms in the noisy setting.

4.4 Numerical experiments – impact on optimization algorithms

In this section, we describe two experiments in which the estimate of the gradient obtained with different schemes is used inside unconstrained optimization algorithms – Gradient Descent with a decreasing stepsize and the Nonmonotone stabilization algorithm – when function evaluations are affected by noise.
In both experiments, we consider ten functions from the Schittowski test set [120] that are associated to the test problems with the following IDs : (201, 203, 205, 208, 256, 271, 275, 276, 290, 291).
The decision to consider only a portion of the functions is based on two key factors: the first, and most important, is the significant computational effort necessary to execute the experiment (almost 385 hours of computation on the set of selected functions). The second reason is that, because this is a preliminary study, we wanted to investigate and display the results for each objective function, which can only be done if the set of functions is limited in size.
We consider these experiments as a first attempt to understand the behaviour of different gradient approximation schemes in optimization algorithms. More extended

experiments - in terms of functions considered, replicas and tested approximation schemes - will be run in future studies.
In Table 4.9 we show the features of the objective functions involved in the experiment. Each row represents a different objective function and the following data are reported in the columns: the dimensions n of the function; the value of the function $f(x^0)$ at the suggested starting point and the value of the true global minima $f(x^*)$ as indicated in [120].

Table 4.9. Features of the objective functions

function	**n**	$\boldsymbol{f(x^0)}$	$\boldsymbol{f(x^*)}$
201	2	45	0
203	2	0.52	0
205	2	14	0
208	2	24	0
256	4	215	0
271	6	750	0
275	4	33	0
276	6	38	0
290	2	9	0
291	10	3025	0

4.4.1 Gradient Descent with a decreasing stepsize

In the first experiment, we consider the Gradient Descent algorithm described in Algorithm 2 (Chapter 2) where the value of the stepsize is determined according to the following rule:

$$\alpha^k = \min\left(\frac{1}{1+k},\ \frac{1}{||\nabla f(x^k)||}\right) \tag{4.23}$$

We decided to use the stepsize (4.23) instead of the one described in (2.8) because we noticed that, with the latter, there were several occasions in which the minimization of a function would fail for numerical issues. This would happen when, in the noisy setting, the stepsize (2.8) caused abrupt movements along the poorly estimated direction d^k, yielding in the following iteration to numerical issues during the computation of the objective function or of the derivatives. This was especially true when the inaccurate estimated direction d^k had a large norm.
The issue could be solved by changing the stepsize (2.8) with

$$\alpha^k = \frac{\alpha^0}{1+\alpha^0\, k} \tag{4.24}$$

and by tuning the parameter α^0. Setting ad hoc values of α^0 for different functions and for different magnitude of the noise level λ would in fact solve this issue.

To avoid performing this fine-tuning for all the functions and noise levels, we opted for the choice of the stepsize as in (4.23) that performs a sort of autotuning without requiring manual intervention. In fact, when the norm of the gradient assumes extremely large values because poorly estimated, the second term becomes smaller than the first one, thus forcing the algorithm to perform an iteration with a small stepsize α^k. With this choice, we did not encounter any numerical issue with any of the considered functions.
We also changed the original stopping condition of the Gradient Descent algorithm, $\nabla f(x) = 0$, with the following:

$$||\nabla f(x)|| \leq 10^{-2}, \tag{4.25}$$

and we added a condition for which the algorithm stops when it reaches a predetermined *maximum number of iterations* without satisfying (4.25).
Algorithm 8 shows the Gradient Descent algorithm with this implementation choices.

Algorithm 8 Gradient Descent algorithm with decresing stepsize

Data $x^0 \in \mathbb{R}^n$, $\alpha^0 > 0$, $k = 0$ and the maximum number of iterations $R > 0$
for $k = 0, \dots, R$ **do**
    Set $d^k = -\nabla f(x^k)$ and $\alpha^k = \min\left(\frac{1}{1+k}, \frac{1}{||\nabla f(x^k)||}\right)$
    Set $x^{k+1} = x^k + \alpha^k d^k$
    if $||\nabla f(x^{k+1})|| \leq 10^{-2}$ **then**
        stop
    end if
end for

In this experiment, we run Algorithm 8 on the set of selected functions replacing $\nabla f(x^k)$ with the estimate of the gradient obtained with the following set of gradient approximation schemes: *FFD*, *CFD*, *GSG*, *cGSG*, *NMXFD*. [2]

We now summarize the settings of the experiment:

- the following values of the standard deviation of the noise λ: $(10^{-3}, 10^{-2})$ and the following values of the stepsize σ that is shared across all the schemes: $(1, 10^{-1}, 10^{-2}, 10^{-3})$ are considered;
- similarly to Sect. 4.3.2, 100 replicas for each combination of (approximation scheme, f, λ, σ) are executed because of the randomicity that is due to the presence of the noise;
- the methods *GSG*, *cGSG*, and *NMXFD* are given $N = 8n$ function evaluations to build the gradient estimate at each iteration;

[2] Inspired by [17], we let the stopping criterion be determined by the evaluation of the *true* gradient norm, rather than the one of the estimated gradient.

- the maximum number of iterations R is set to 5000 [3] for all the functions;

Tables [4.11, 4.12] show the results obtained with different values of λ. On the columns we reported the median value of the objective function in correspondence of the final point $\bar{x}$ across 100 replicas and the median number of iterations needed by the algorithm to reach $\bar{x}$. In each row we report the values associated to the *best* stepsize for each method. To clarify this point, consider the example of Table 4.10: we can see that the performances of *NMXFD* on function 208 differ as the value of the stepsize σ changes. To avoid reporting four rows for each combination of (scheme, function) – the resulting table would be gigantic – and to avoid cherry picking the value of σ (e.g. $\sigma = 0.001$ yields to the best performance of *NMXFD* on this function, but $\sigma = 0.01$ is the one for which *GSG* perform better), we decided to display only *one* row for each (function, scheme), the one associated to the best performance in terms of minimum value of *median* f^* and highlighted in bold in the table. For each function, we highlight in bold the best value of objective function and the lowest number of iterations and we italicize the second best.

Table 4.10. Example with NMXFD - function 208

function	**scheme**	$\boldsymbol{\sigma}$	$\boldsymbol{f^*}$	**iterations**
208	NMXFD			
		0.001	**0.006**	**5000**
		0.01	0.01	5000
		0.1	0.62	5000
		1	0.95	5000

We can notice that, despite the noise, for almost all the functions the differente schemes manage to reach points whose objective function value is close to the one of the global minimum of the function [4]. This confirms the goodness of the proposed stepsize (4.23) since it mitigates the negative effects of iterations with poorly estimated gradients.
A part from the *FFD* method that seems to generally perform worse than the other ones, it is not possible to identify a clearly better performing scheme from this experiment.
With respect to the *GSG* method, for $\lambda = 10^{-3}$, we have that the *NMXFD* scheme performs better – either in terms of value of the objective function or on the number of iterations needed to reach the final point – on six functions, equal on three and there is only one function where it performs worse.
For the same value of $\lambda = 10^{-3}$, *NMXFD* performs better than *cGSG*in five occasions, equal in three and worse in two of them.

[3] Because *FFD* and *CFD* use $N = n+1$ and $N = 2n$ function evaluations to compute the gradient estimate, we set for them R to 40000 and 20000, respectively, to ensure that each scheme has the same budget of function evaluations in the experiment.

[4] The value 0 in the tables indicates values that are smaller than 10^{-4}.

Table 4.11. Algorithm 8 with $\lambda = 10^{-3}$

function	scheme	f^*	iter	function	scheme	f^*	iter
201				**271**			
	FFD	**0**	82		FFD	**0**	40000
	CFD	**0**	**51**		CFD	**0**	**189**
	GSG	**0**	*56*		GSG	**0**	5000
	cGSG	**0**	*56*		cGSG	**0**	*194*
	NMXFD	**0**	**51**		NMXFD	**0**	**189**
203				**275**			
	FFD	**0**	40000		FFD	**0.012**	40000
	CFD	**0**	*20000*		CFD	*0.015*	20000
	GSG	**0**	**5000**		GSG	0.023	**5000**
	cGSG	**0**	**5000**		cGSG	0.021	**5000**
	NMXFD	**0**	**5000**		NMXFD	0.022	**5000**
205				**276**			
	FFD	**0**	22305		FFD	**0**	40000
	CFD	**0**	*20000*		CFD	**0**	**2770**
	GSG	**0**	**5000**		GSG	0.001	5000
	cGSG	**0**	**5000**		cGSG	0.001	5000
	NMXFD	**0**	**5000**		NMXFD	**0**	*2862*
208				**290**			
	FFD	0.04	40000		FFD	**0**	16
	CFD	**0**	*20000*		CFD	**0**	14
	GSG	*0.01*	**5000**		GSG	**0**	**10**
	cGSG	*0.01*	**5000**		cGSG	**0**	13
	NMXFD	*0.01*	**5000**		NMXFD	**0**	*11*
256				**291**			
	FFD	**0**	40000		FFD	**0**	164
	CFD	**0**	*210*		CFD	**0**	66
	GSG	**0**	5000		GSG	**0**	84
	cGSG	**0**	727		cGSG	**0**	**57**
	NMXFD	**0**	**177**		NMXFD	**0**	*62*

Table 4.12. Algorithm 8 with $\lambda = 10^{-2}$

function	scheme	f^*	iter	function	scheme	f^*	iter
201				**271**			
	FFD	**0**	463		FFD	0.005	40000
	CFD	**0**	**52**		CFD	**0**	219
	GSG	**0**	98.5		GSG	*0.001*	5000
	cGSG	**0**	*55*		cGSG	**0**	*196*
	NMXFD	**0**	**52**		NMXFD	**0**	**191**
203				**275**			
	FFD	**0**	40000		FFD	**0.012**	40000
	CFD	**0**	18035		CFD	*0.015*	20000
	GSG	**0**	*5000*		GSG	0.023	**5000**
	cGSG	**0**	*5000*		cGSG	0.023	**5000**
	NMXFD	**0**	**3480**		NMXFD	0.022	**5000**
205				**276**			
	FFD	**0**	40000		FFD	0.002	40000
	CFD	**0**	7777		CFD	**0**	**2767**
	GSG	**0**	**5000**		GSG	*0.001*	5000
	cGSG	**0**	**5000**		cGSG	*0.001*	5000
	NMXFD	**0**	**5000**		NMXFD	**0**	*2861*
208				**290**			
	FFD	0.04	40000		FFD	**0**	15
	CFD	**0**	20000		CFD	**0**	14
	GSG	*0.01*	***5000***		GSG	**0**	**9**
	cGSG	*0.01*	***5000***		cGSG	**0**	*11*
	NMXFD	*0.01*	***5000***		NMXFD	**0**	*11*
256				**291**			
	FFD	**0**	40000		FFD	**0**	19586
	CFD	**0**	*211*		CFD	**0**	65
	GSG	**0**	5000		GSG	**0**	83
	cGSG	**0**	737		cGSG	**0**	**54**
	NMXFD	**0**	**178**		NMXFD	**0**	*63*

With *CFD* there is almost a perfect tie. The two methods have better performances on three functions each, and have the same performances on four functions.
When the noise level increases, *NMXFD* performs better than *GSG* on seven functions and worse on one, and it performs better than *cGSG* on six functions and worse on one. Finally, with respect to *CFD*, *NMXFD* reaches better solutions or requires less iterations to reach the same points in six occasions, performing worse on three functions.
It appears that the *NMXFD* scheme performs at least as well as the other ones, and that it shows improvements as the noise level increases. Because of the small number of functions considered, and because the differences between the different methods are often minor, both in terms of the value of the objective function found and the number of iterations required to reach the solution, these conclusions should be carefully pondered. Nonetheless, it appears that the *NMXFD* method is never outperformed by the other schemes, making it a viable method for estimating gradients in an optimization framework.
In the next section, we describe the numerical experiments where the estimate of the gradient is used inside the Nonmonotone stabilization algorithm.

4.4.2 Nonmonotone stabilization algorithm

In this experiment, we consider the NonMonotone Stabilization Algorithm 2 described in Sect. 2.4.3.
Similarly to the experiment described in the previous section, we run the algorithm on the set of functions shown in Table 4.9, replacing $\nabla f(x^k)$ with the estimate of the gradient obtained with the following set of gradient approximation schemes: *FFD*, *CFD*, *GSG*, *cGSG*, *NMXFD*.
In particular, we implemented the NMS2 algorithm with the following implementation choices:

- we set the number of iterations to 5000 [5];
- we set N, that represent the "budget" allocated to the exploration of the BB stepsize, to 10;
- we set M, the "memory term" that appears in Condition 2, to 5;
- we use the stopping criterion $||\nabla f(x)|| \leq 10^{-2}$ based on the value of the *true* gradient;

The setting of the experiments is almost identical to the one described in Sect. 4.4.1, and is reported here again for the convenience of the reader:

- we consider the values of λ: $(10^{-3}, 10^{-2})$ and the values of the stepsize σ: $(1, 10^{-1}, 10^{-2}, 10^{-3})$;

[5] With the same reasoning of Sect. 4.4.1, we set this value to 40000 and 20000 for the *FFD* and *CFD* methods, respectively

- we provide the methods *GSG*, *cGSG*, and *NMXFD* with $N = 8n$ function evaluations to build the gradient estimate;
- we run 100 replicas for each combination of (approximation scheme, f, λ, σ) and we report the median values of the results.

Tables [4.13, 4.14] show the results obtained for the values of λ 10^{-3} and λ 10^{-2}, respectively. For each function, we report the results obtained by each scheme with the value of σ that yields to the solutions with the lowest value of the objective function.
As a general comment, we can observe without surprise that this algorithm is much faster than the gradient descent with a decreasing stepsize regardless of the scheme used for the gradient estimate.
In terms of the performance of the various methods, we can see from Table 4.13 that the *NMXFD* scheme is faster than *GSG* at finding the optimal solution in six instances, and is slower two times. When comparing the method to the *cGSG* scheme, the situation is similar, with the method performing better on seven occasions and worse only once. We treated function *205* as a draw because *NMXFD* finds the solution after only three iterations, but the objective function value is slightly higher than that of the other methods.
The comparison with the *CFD* is more balanced, with the *NMXFD* method being faster on four occasions and slower on two. Let us note that, despite the higher number of iterations provided to the *CFD* scheme, it manages to satisfy the stopping criterion way before reaching the maximum number of iterations, thus using a fewer number of function evaluations than the *NMXFD* scheme overall.
The experiment with the level of noise $\lambda = 10^{-3}$ therefore suggests that, if the user trying to minimize a function wants to perform as few function evaluations as possible, he could use the *CFD* method. If his interest is in reaching the optimal solution in a fewer number of *iterations* possible, than considering other schemes, included the *NMXFD*, can bring benefits. It is also worth noting that, because *NMXFD* is a scheme with different parameters, it always possible to set $N = 2n$ as the number of function evaluations used to build the gradient estimate, thus obtaining the same exact performance as the *CFD* method.
As the noise level increases to $\lambda = 10^{-2}$, the *NMXFD* scheme performs better than both the *cGSG*, *GSG* and *CFD* schemes on seven functions and worse on one.

4.5 Discussion and and future works

In this chapter, a novel scheme to estimate the gradient of a function is proposed. It is based on linear functionals defining a filtered version of the objective function. Unlike standard methods for which the approximation error can only be characterized from a statistical point of view, one advantage of the proposed scheme relies on a deterministic characterization of the approximation error in the noise-free setting.

Table 4.13. Nonmonotone stabilization algorithm with $\lambda = 10^{-3}$

function	scheme	f^*	iter	function	scheme	f^*	iter
201				**271**			
	FFD	**0**	148		FFD	*0.001*	170
	CFD	**0**	**10**		CFD	**0**	**9**
	GSG	**0**	112		GSG	**0**	147
	cGSG	**0**	40		cGSG	**0**	*24*
	NMXFD	**0**	*11*		NMXFD	**0**	**9**
203				**275**			
	FFD	*0.001*	74		FFD	0.002	193
	CFD	**0**	118		CFD	**0**	*18*
	GSG	**0**	*45*		GSG	*0.001*	123
	cGSG	**0**	**35**		cGSG	*0.001*	61
	NMXFD	**0**	99		NMXFD	**0**	**17**
205				**276**			
	FFD	*0.001*	156		FFD	*0.001*	148
	CFD	**0**	175		CFD	**0**	*7*
	GSG	**0**	*105*		GSG	*0.001*	97
	cGSG	**0**	118		cGSG	*0.001*	46
	NMXFD	*0.001*	**3**		NMXFD	**0**	**6**
208				**290**			
	FFD	*0.002*	**77**		FFD	**0**	*11*
	CFD	**0.001**	163		CFD	**0**	**6**
	GSG	**0.001**	*141*		GSG	**0**	16
	cGSG	**0.001**	163		cGSG	**0**	16
	NMXFD	**0.001**	146		NMXFD	**0**	**6**
256				**291**			
	FFD	0.003	62		FFD	*0.002*	137
	CFD	**0**	*50*		CFD	**0**	**24**
	GSG	*0.002*	186		GSG	**0**	108
	cGSG	**0**	291		cGSG	**0**	33
	NMXFD	**0**	**45**		NMXFD	**0**	*27*

Table 4.14. Nonmonotone stabilization algorithm with $\lambda = 10^{-2}$

function	scheme	f^*	iter	function	scheme	f^*	iter
201				**271**			
	FFD	0.005	183		FFD	0.008	166
	CFD	**0**	**9**		CFD	**0**	*13*
	GSG	*0.002*	168		GSG	0.003	151
	cGSG	**0**	40		cGSG	**0**	25
	NMXFD	**0**	*10*		NMXFD	**0**	**10**
203				**275**			
	FFD	0.004	*71*		FFD	0.007	177
	CFD	*0.002*	133		CFD	**0.001**	*31*
	GSG	*0.002*	73		GSG	*0.004*	166
	cGSG	**0.001**	**60**		cGSG	**0.001**	65
	NMXFD	*0.002*	96		NMXFD	**0.001**	**22**
205				**276**			
	FFD	0.004	179		FFD	0.003	149
	CFD	*0.003*	192		CFD	**0**	*10*
	GSG	**0.002**	*152*		GSG	0.002	161
	cGSG	**0.002**	163		cGSG	*0.001*	50
	NMXFD	*0.003*	**2**		NMXFD	**0**	**9**
208				**290**			
	FFD	0.008	**95**		FFD	**0**	30
	CFD	**0.002**	168		CFD	**0**	**6**
	GSG	0.005	*158*		GSG	**0**	18
	cGSG	*0.003*	174		cGSG	**0**	*15*
	NMXFD	**0.002**	144		NMXFD	**0**	**6**
256				**291**			
	FFD	0.01	89		FFD	0.013	126
	CFD	**0**	*135*		CFD	**0**	**28**
	GSG	0.004	210		GSG	*0.002*	176
	cGSG	*0.001*	243		cGSG	**0**	*32*
	NMXFD	**0**	**123**		NMXFD	**0**	**28**

The other advantage lies in its behavior when function evaluations are affected by noise. In fact, the variance of the estimation error of the proposed method is showed to be strictly lower than that of the Central Finite Difference scheme and diminishes as the number of function evaluations increases. The suitable linear combination of finite differences seems to have a filtering role in the case of noisy functions, thus resulting in a more robust estimator.
Numerical experiments on a significant benchmark given by the 69 Schittkowski functions show the good performances of the proposed method when compared with those of the standard methods commonly used in the literature. In particular, the performances of NMXFD are comparable with those of CFD in absence of noise and better with noisy data and seem to be better than those of other schemes in both scenarios. Moreover, we also show the comparison with NMXFD and the average of repeated CFD, thus using the same budget of function evaluations. As the noise level increases, NMXFD tends to perform better than all the other schemes.
This supports the idea that the theory developed to propose this new scheme can be a suitable framework to design gradient estimates with noisy data. The gradient estimate proposed in this chapter can be seen as a first design attempt. A future study could be dedicated to the investigation of the best gradient estimates in this framework.
The results obtained with the preliminary experiments described in Sect. 4.4.1 and 4.4.2 suggest that the *NMXFD* method is indeed a valid method for estimating gradients in an optimization framework when function evaluations are affected by noise. In a one-to-one comparison with all the schemes that use the same budget of function evaluations to build the gradient estimate, it outperforms them in most of the occasions.
The level of noise used in the experiment had a limited magnitude, the standard deviation of the noise λ assuming values 10^{-3} and 10^{-2}.
In other works that explore the impact of noise in the resolution of optimization problems, similar magnitudes of noise have been used. For example, in [17] the additive noise used in the numerical experiments assumes the following possible values: [0, 10^{-4}, 10^{-2}]. In the numerical experiments in [15], where the noise is modelled as a uniform random variable $\epsilon(x) \sim \mathcal{U}(-\xi, \xi)$, the following value of ξ are considered: $[10^{-8}, 10^{-6}, 10^{-4}, 10^{-2}]$.
It therefore appears that the impact of the noise on the resolution of optimization problem has been studied by the research community in the the cases where the value of the noise is limited in magnitude. The *CFD* is almost unbeatable when the noise is zero or close to zero, and still represents a valid scheme when the noise level is small. It is now perhaps the moment to start research activities that expand the analysis to situations in which the value of the noise is larger than the ones considered so far.
When the level of the noise increases significantly, it becomes harder to come to conclusions about the impact of the different gradient approximation schemes. In

fact, it often happens that the minimization of different functions fails when the noise level is high. Often, the reasons behind the failing are independent of the quality of the gradient estimate: the value of the stepsize, whose "correct" value is hard to determine in presence of noise, might be the responsible of the failing; the optimization algorithm could fail to reach an optimal point if provided with a small number of maximum iterations; the value of the noise could be so high in some points that the gradient estimates produced by the different methods are nothing but random guesses. These are perhaps some of the reasons why, at the best of our awareness, high level of noise have not been studied extensively.
A future study could therefore be dedicated to the investigation of the impact of different gradient estimates on the performance of optimization algorithms when high levels of noise are present.

Let us now describe a preliminary experiment that gave us some ideas for other potential future works. Let us consider the function *276* that has also been used in the previous experiments. We tried to minimize this function using the Gradient Descent algorithm described in Sect. 4.4.1, setting $\sigma = 0.1$ and running 100 replicas of the experiment for the noise levels $\lambda = 10^{-1}$ and $\lambda = 1$.
The median value of the objective function and of the number of iterations required to reach the solution are reported in Table 4.15.

Table 4.15. Algorithm 8 - function 276

	scheme	$\boldsymbol{f^*}$	**iter**
$\lambda = 0.1$			
	FFD	0.005	40000
	CFD	*0.001*	18223
	GSG	*0.001*	*5000*
	cGSG	*0.001*	*5000*
	NMXFD	**0.000**	**3144**
$\lambda = 1$			
	FFD	0.066	40000
	CFD	0.031	*20000*
	GSG	*0.008*	**5000**
	cGSG	**0.005**	**5000**
	NMXFD	0.016	**5000**

From the table it is possible to see that the *NMXFD* scheme performs better than the other schemes for $\lambda = 10^{-1}$ and that, for an higher value of λ it finds a better solution than the *CFD* scheme and a worse one than the *GSG* and *cGSG* schemes. We were a little surprised by this result since, for the same function, the *NMXFD* scheme had performed better than the Gaussian Smoothing schemes for both $\lambda = 10^{-3}$ and $\lambda = 10^{-2}$, and the results of the numerical experiments of Sect. 4.3 showed a better

behaviour of the *NMXFD* scheme on the quality of the produced gradient estimates also for higher values of the noise. Trying to better understand what happened as the noise level increased, we designed this small experiment:
we minimize function *276* with Algorithm 8 using the gradient estimate obtained with the *cGSG* scheme. At each iteration, we save the *real* gradient and the gradient estimate obtained with *cGSG*, as well as the *real* value of the objective function and the gradient estimate that we would have obtained using the *NMXFD* scheme from the same point. In other words, at each iteration we start from the same point x^k, we build the gradient estimate with both the methods, we save quantities that measure the quality of the estimate for *both* the schemes, then we generate the point x^{k+1} according to the *cGSG* scheme and we move to the next iteration. The idea behind this was that of trying to answer the question: what would have happened if we had used the estimate of *NMXFD* at each iteration instead of the *cGSG* one?
We found something that we think is worth investigating in future works: at the beginning of the minimization procedure the *NMXFD* method find noticeably better gradient estimates (the *angle* between the real and estimated gradient is smaller, and the objective function of the point that would be reached using the estimated direction has a lower value) than the *cGSG* scheme. After some iterations though, the *NMXFD* seems to lose its advantage, finding better estimates only around half of the times.
We noticed that the number of iterations after which this starts to happen, depends on the noise level. As the noise level increases, the *NMXFD* methods starts to lose its advantage over the *cGSG* more rapidly.
For $\lambda = 10^{-3}$ the *NMXFD* method produces better estimates of the gradient in *100%* of the first 15 iterations. For $\lambda = 10^{-2}$ this happens *80%* of the times, and it happens *62%* and *37.5%* of the times for $\lambda = 10^{-1}$ and $\lambda = 1$, respectively. Over the entire 5000 iterations, the amount of times in which *NMXFD* method finds better estimates than its counterpart ranges between *34.4%* and *49.27%* for the four different values of the noise that we have considered.
We saw the same behaviour also for a couple of other functions that we tried experimenting with.
In future works, it would be interesting to try to identify the threshold after which the advantage of *NMXFD* fades. This threshold could be computed as the ratio between the noise level and other quantities (e.g. the value of the objective function, or the norm of the gradient), but this is only an idea. It would also be interesting to study if different methods can interact with each other to generate better results.
We report in Table 4.16 an example of the data produced with this experiment consisting of the first 15 iterations of the algorithm with $\lambda = 10^{-1}$. The first column indicates the iterations, the second and third column the angle between $\nabla f(x^k)$ and the estimated obtained with the *NMXFD* and *cGSG* schemes, respectively. In the last two columns we reported the ratio between the value of λ and the values of $f(x^k)$ and $||\nabla f(x^k)||$, two potential quantities that can be used to determine the

threshold discussed before.

Table 4.16. Details of the preliminary experiment

k	NMXFD angle	cGSG angle	$f(x^k)$	$\|\|g(x^k)\|\|$	$\lambda/f(x^k)$	$\lambda/\|\|g(x^k)\|\|$
1	**2.1**	24.8	38.140	15.700	0.003	0.006
2	**4.7**	16.9	25.268	12.745	0.004	0.008
3	**4.6**	16.0	14.551	9.654	0.007	0.010
4	**5.7**	33.0	6.763	6.549	0.015	0.015
5	**10.4**	35.0	2.485	3.797	0.040	0.026
6	**26.1**	26.6	0.547	1.509	0.183	0.066
7	44.5	**33.6**	0.360	1.067	0.278	0.094
8	**36.5**	42.5	0.290	0.868	0.345	0.115
9	78.7	**17.6**	0.238	0.709	0.419	0.141
10	76.7	**19.5**	0.176	0.477	0.567	0.209
11	**28.7**	37.4	0.153	0.409	0.653	0.244
12	67.6	**38.3**	0.137	0.363	0.731	0.275
13	**35.1**	65.5	0.128	0.352	0.783	0.284
14	**59.1**	69.3	0.123	0.353	0.811	0.283
15	144.5	**17.3**	0.121	0.344	0.828	0.291

Chapter 5

Simulation-Based Optimization: a neural network approach

After introducing the Emergency Department (ED) problem, often addressed with the so called *Simulation-Based Optimization* (SBO) approach, this chapter will focus on the description and application of an approach to solve SBO problems that involves the use of Neural Networks as an approximating model of the objective function obtained by the simulation software.
This approach, that enables the usage of gradient-based methods for the resolution of the optimization problem, is justified by the fact that function evaluations are significantly expensive and that the only time-consuming phase of the neural network approach is in the generation of the dataset which can be parallelized over whatever number of processing systems at disposal.
We compare the performance of the neural network approach with the results obtained by a globally convergent Derivative Free algorithm.

The Chapter is organized as follows. Sect. 5.1 introduces the ED problem. In Sect. 5.2 a review of the literature devoted to ED management is reported, highlighting the most commonly adopted approaches for studying ED operation. Sect. 5.3 deals with the formal statement of the MIU optimal resource allocation problem and Sect. 5.4 describes the methodologies we propose for tackling it, that include an approach based on the usage of artificial neural networks. In Sect. 5.5 a real case study is reported and the related MIU management problem is stated in Sect. 5.6. Experimental results on the case study are included in Sect. 5.7. Finally, in Sect. 5.8 some concluding remarks are drawn.

5.1 Introduction

Emergency Medical Services play a central role among healthcare services since they are devoted to provide timely medical treatments to people needing urgent care [8]. In particular, Emergency Department (ED) is usually considered the most important,

since hospital EDs usually provide the first care to urgent patients transported by ambulance or arriving autonomously. Unfortunately, the worldwide phenomenon of the overcrowding may significantly affect the quality and promptness of the care delivered. Indeed, this phenomenon often gives rise to long waiting times that may endanger the life of critical patients by increasing the risk of deterioration in the health conditions. Besides visit and treatment delay (especially for low acuity patients), overcrowding generates unpleasant phenomena such as an excessive number of patients in the ED, patients treated in the hallways, an increasing number of patients who leave without being visited (LWBS), ambulance diversion, reduced patient satisfaction, and overloaded ED staff. Moreover, even an increase of patient mortality can be directly related to the ED crowding problem.
Causes of overcrowding can be traced back to both exogenous factors as flue season, requests of non–urgent visits and endogenous factors as ED internal resources shortage (observation units, beds), undersized staffing.
Several Key Performance Indicators (KPIs) can be adopted to assess ED performance (see the recent paper [131]) and, in particular, to estimate the level of overcrowding. The most common are the Length Of Stay (LOS) in the ED, the waiting time before the first medical visit (Door-to-Doctor time – DTDT), the percentage of LWBS patients. Moreover, some more elaborate measures have been proposed to quantify crowding and staff workload in an ED in order to possibly prevent the ED from reaching critical conditions by adopting solutions in advance. Some examples are NEDOCS, READ, EDWIN, and the Work Score, [3, 18, 133–135]; however, none of them proved to be able to timely provide warnings at low rates of false alarms [65].
An efficient management of the ED resources (both physical and human ones) along with the adoption of strategies to better handle patient flow trough an ED is at the basis of any attempt to tackle the overcrowding problem. In particular, specific Fast–Tracks (FTs) are often adopted to reduce this phenomenon. FTs consist in separate specific pathways for patients with non–urgent complaints and uncomplicated diseases, identified by the triage nurse and directed to a dedicated unit or area, where they are treated and possibly discharged in short time.
FTs were introduced in the late 1980s in North American hospitals and later in the United Kingdom and Australia. Today they are adopted in hospitals around the world, and the advantages of using FTs have been highlighted in many case studies (see, e.g., the review article [141]). A significant reduction of the ED LOS is usually observed as well as a decrease of the percentage of LWBS patients and of the average number of patients in the ED. As a direct consequence, more timely care can be delivered along with an higher quality of care and hence an improved patient satisfaction and safety are obtained. This motivates the rapid spread of FTs, which nowadays are often adopted in most of the EDs. However, one of the main issues to be considered when creating FTs is an efficient allocation of the resources required by the units devoted to receive FT patients. Indeed, such units, generally called Minor Injuries Units (MIUs), require: a dedicated staff, usually senior medical and

nursing personnel, not shared with other ED units, that can possibly make quick discharge decision; a number of observation units, namely armchairs, stretchers and beds to be used for patient stay. These human and physical resources must be defined in advance, as well as the MIU opening hours. In fact, MIUs usually provide a diurnal service whose working hours commonly coincide with greatest overall ED patient daily influx. Of course, the usage of such resources strictly depends on the percentage of patients directed to the FT by the triage nurse. Note that too prudent behaviour of the triage nurse, namely a conservative attitude to maintain within the ED most of the patients, would nullify the FT goal, so that an enhanced triage should be associated with the adoption of FT.
Of course, management costs of a MIU are related to the quantity of allocated resources and to the opening hours. Therefore, taking into account the variability in time of operations and the constraints on the availability of dedicated personnel and specific observation units, the use of a Decision Support System (DSS) should be desirable to allow ED managers to efficient manage the FT. As far as the authors are aware, such a specific DSS system is rarely adopted and managers usually rely on their own experience to decide the resources to be allocated for the MIU and its working hours. Note that the same reasoning applies also for specialist clinical pathways (like ophthalmologist, orthopedist ones and many others), which are often employed in EDs to direct patients with specific pathologies (clearly identifiable during the triage) straight to a specific unit.
In this paper we deal with MIU optimal resource allocation problem as a strategy to manage ED low–complexity patients within a FT system. In particular, we formulate this problem as a Multiobjective Optimization (MO) problem [1] aiming at minimizing both the expected value of the overall patient waiting time (FT and non-FT patients) and MIU management costs, measured in terms of working hours of the MIU. Note that these are two conflicting goals, since by increasing MIU working hours (which implies costs grow), a reduction of patient waiting time is expected. This approach can be easily generalized by considering further objectives which reflect other resource usage, too.
A major issue concerns such an MO optimization problem: it cannot be stated in analytical form, since the KPIs of interest, i.e. patient waiting times and costs, are not available in closed form. In fact they cannot be computed by means of an analytical function of the variables which represent the ED settings. It is imperative to resort to a simulation model representing the ED operation, obtaining the KPIs of interest as response of simulation runs. Therefore, actually, the problem at hands belongs to the class of Simulation–Based Optimization (SBO) problems and, as a direct consequence, it is normally solved using black–box and derivative–free optimization methods.
In this work, we reduce the MO problem to a series of single objective problems with the scalarization approach described in Appendix C.

[1] We refer the reader to Appendix C for some brief notes on MO.

When solving each problem, runs of a simulation model are used for evaluating objective and constraints functions of the problem and the overall optimization procedure results very computational expensive. This drawback is even more serious whenever a high number of replications in the simulation runs is necessary to ensure accurate responses of the simulation model. This is a well known big minus, common to SBO approaches.
To overcome this drawback, in this paper we propose a novel approach based on the use of an Artificial Neural Network. In particular, we build a machine learning model which allows us to evaluate an approximation of the problem functions without resorting to a simulation model. More precisely, the simulation model is only used for generating a dataset needed for training a Multi–Layer Perceptron network under the supervised learning paradigm. Then, at each iteration of the optimization algorithm, function evaluations are performed by means of the machine learning model, in place of the simulation model.
Besides a great reduction of the computational time, this new approach has some important advantages: gradient–based methods can be adopted for solving the MO problem, since the approximate model consists of continuously differentiable functions; parallel computing techniques can be adopted for further improving the overall efficiency; additional constraints, not involving KPIs computed via the simulation model, can be possibly introduced in a simple way, avoiding re–building the model from scratch.
A real case study is considered for testing the novel methodology we propose. In particular, the operation of the ED of a large Italian hospital in Rome, Italy is studied, focusing on the MIU optimal resource allocation problem. After formulating the problem on the basis of the available data, both the approach based on simulation–based MO derivative–free optimization and that based on a artificial neural network are experimented and the obtained results compared. The results show that the latter approach is promising, not only because it allows to significantly reduce the computation time needed to obtain the Pareto front, but also because it allows for the generation of more non-dominated points that are better distributed than those obtained with the simulation–based MO derivative–free optimization approach.

5.2 Literature review

The literature dedicated to the ED overcrowding is really very wide. The reader is referred to the review paper (and to the references reported therein) [64], where the main overcrowding causes and effects are highlighted along with some possible solutions. Some other examples of papers dealing with the overcrowding phenomenon are [39, 67, 94, 104, 106, 138].
An efficient management of the resources is usually indicated as possible tool for reducing ED overcrowding. Analytical methods and modeling approaches for achieving an optimal resource allocation are reviewed in [5]. This paper points out

that strategies like fast tracking and enhanced triage models are commonly adopted against the overcrowding, requiring additional resources. The review article [141] discusses strategies like team triage, streaming and fast tracking that have proven to reduce ED overcrowding along with some ideal ED patient journey models. It is important to remark that diverting patients with low–acuity illnesses and injuries is a strategy analyzed in many studies aiming at assessing its impact on reducing the waiting times and, accordingly, the ED overcrowding [33, 68, 70, 93, 104, 121]. Generally speaking, the aim of adopting such a diversion policy, like the FT system, is to reduce the ED overcrowding by discharging the patients earlier. In particular, [104] points out that the expected benefits of using FT systems are observed in every ED, regardless of the single case studies considered. In [121], the authors show that reducing the number of low–complexity patients does not significantly affect the waiting times of the high–complexity patients. This result is supposed to hold when the resources in charge of the visit and treatment of low–acuity patients are dedicated. Contrarily, when such resources are shared with critical patients, a worsening in the total average waiting times of the most urgent patients may be experienced [77]. Moreover, the authors in [77] show that most benefits of the FT system are observed in EDs with a considerable number of urgent patients. Indeed, the improvement in terms of waiting time is higher when the low–complexity patients can bypass a larger number of patients. In the interesting case study reported in [1], the effect of FT services on ED performance of Australian hospital EDs are clearly analyzed, concluding that FTs enable providing an effective care to patients with minor illnesses without negatively affect treatments on other non-FT patients. The results of another case–controlled study are reported in [32], where the several benefits of adopting FT are clearly highlighted.

However, most of the published papers use simulation modeling to assess the effectiveness of the diversion policy towards FT. Sometimes, simulation is combined with other techniques (see, e.g., [62] where simulation is combined with the system dynamics approach) but, to the best of our knowledge, papers using optimization are still very few. Indeed, many papers examine the impact of the adoption of these strategies through scenario–based analyses and only a few of them aim at finding the settings providing an effective diversion policy by optimizing some KPIs. For instance, in [123], a multi-criteria method is proposed to determine the best configuration for the FT system in terms of five performance indicators. In [38], a mixed–integer linear programming problem is proposed to minimize the number of waiting patients by focusing on the management of human and material resources. In [4], the final goal is to assess how different staffing levels affect the performance in terms of patient throughput when budget restrictions are considered; for this purpose, a discrete stochastic optimization problem concerning an ED in Kuwait is considered. In [144], the total average patient waiting time is minimized through a simulation–based metamodeling approach, which is used to determine the optimal ED resource allocation by considering also budget and capacity constraints; the computationally

expensive simulation model is replaced by a suitable metamodel, chosen among several alternatives. Recently, [142] has dealt with a resource allocation problem by analyzing the impact of human errors due to the increase in workload within the EDs; the approach proposed in this paper combines several techniques, such as simulation and artificial neural networks.

As regards approaches commonly used for analyzing ED operation, several Operations Research methodologies are usually adopted, namely Analytical Models [116], Statistical Analysis [41, 71] and Simulation. Simulation turns out to be the best suited for dealing with the complex and stochastic processes emerging in healthcare systems, like the ED (see [24] for an analysis of the literature related to simulation and modeling in the healthcare contexts). In particular, Discrete Event Simulation (DES) (see, e.g., [14, 45, 58, 68, 76, 78, 139, 144]) and Agent–Based Simulation (ABS) (see, e.g., [7, 70, 80]) are very frequently used for studying patient flow through an ED. We refer the reader to the recent paper [118] (and to the references reported therein) for a complete review on simulation modeling applied to EDs, along with a discussion of current patterns and trends.

Recently, some papers dealing with ED management have proposed approaches combining a simulation model with an optimization algorithm [4, 40, 59, 60], resulting in the so–called Simulation–Based Optimization [49, 53]. Approaches based on SBO allow optimizing the values of some KPIs of interest, hence determining the optimal settings of the system considered. For the state of the art on SBO applied to EDs, we refer the reader to [143]. Since in the SBO approach DES models are used for representing the ED operation, the objective function and the constraints values corresponding to a specific ED setting have to be evaluated by running a simulation model. Therefore, the use of black–box and derivative–free optimization algorithms is mandatory for solving optimization problems within the SBO approach. Furthermore, [143] evidences the frequent need for SBO approaches to address MO problems, which require suited efficient solution algorithms. In [129, 130], the multiobjective genetic algorithm NSGA-II [43] is used to find the optimal configuration of a Swedish ED in order to minimize several objectives, such as DTDT and LOS, meeting the national guidelines. A multiobjective analysis is described in [79] to determine the bed capacity allocation in order for the Italian hospital considered to efficiently manage both elective patients, coming from the waiting lists, and ED patients; the optimization algorithm embedded in the simulation software is based on the simulated annealing method. In [146], a multiobjective tabu search is developed for solving an ED layout problem, where closeness among departments and patient's travel distance and time are the objectives to be minimized. Finally, in [46] and [26], optimal computing budget allocation is combined with genetic algorithm and particle swarm optimization, respectively, to determine the optimal allocation of personnel and medical equipment in order to minimize LOS and resource waste cost.

It is important to remark that the problem formulation as a MO problem has a very important practical advantage: the optimal solution is provided in terms of a set of

non–dominated points, the Pareto front, thus allowing the ED managers to select the best point according to their preferences.

5.3 Problem statement

In order to state the problem we are dealing with, we first briefly recall some structural and operational elements which characterize an ED. As well known, the *triage process* is the first activity a patient is subjected to when arriving in the ED. In this phase, a first assessment is performed by a triage nurse and a *severity tag* is assigned to each patient. Such assignment corresponds to a prioritization of patients on the basis of acuity. Different scales can be adopted; an example is the five–level Emergency Severity Index (ESI) adopted in the USA, providing a stratification of patients into five groups from 1 (most urgent) to 5 (least urgent) [51]. For simplicity, tags are often associated with colors, e.g., red, yellow, green and white, used in a four–level scale (listed according to severity decrease). National Health Systems guidelines usually provide a threshold value that should not be exceeded by average waiting time DTDT and stay time LOS, for each triage tag. We will denote by $\mathcal{T}$ the set of the severity tags.
From the structural point of view, an ED is usually made up of a number of *units* which are ED areas (rooms) where patients are directed after the triage, based on the pathology observed (e.g., surgical unit, resuscitation area, etc). In the sequel we will indicate by U the set of ED units. More precisely, for each severity tag $t \in \mathcal{T}$, we denote by $U(t)$ the set of ED units where a t-tagged patient can be directed. This is necessary to take into account that, being ED units usually different equipped, patients can only be assigned to some of them, on the basis of their triage tag.
We consider the problem of defining FT systems within EDs, aiming at providing prompt and effective care to low–complexity patients. The strategy consists in diverting such patients towards a dedicated ED unit (MIU) in order to visit, treat and discharge them in short time, without crowding standard clinical pathways of the ED. An overall decrease of the number of patients in the ED and the patient waiting times and LOS are expected, and consequently a reduction of ED overcrowding. We remark that, two key issues underlie the success of such a FT strategy:

- a suited allocation of the physical and human MIU resources;
- an enhanced triage, which should allow promptly identifying patients with low–acuity illnesses and injuries to be directed to MIU via FT.

The first issue concerns the resources to be allocated in the MIU. Indeed, MIU usually consists of a number of rooms equipped with armchairs, stretchers and beds where dedicated personnel (physicians and nurses) visit and treat patients. Moreover, since MIU is usually a diurnal service, its opening and closing hours must be defined for each day of a prefixed time period of interest.

The second issue regards the percentage of patients that are directed to the FT by the triage nurse. Of course, nurse decision is based on patient clinical assessment, however a too prudential attitude of the triage nurse would imply a too low percentage of patients diverted to the MIU, hence making the establishment of FTs useless. In this work, we consider percentage of patients that are directed to the FT by the triage nurse as fixed, with the value set to 30%.
We can now formally introduce the problem. Let $D = \{1, \dots, n\}$ be the set of the days of the chosen time period (for instance, $D = \{1, \dots, 7\}$ if a weekly schedule is adopted). We assume that each MIU room is equipped by a prefixed number of observation units (armchairs, stretchers and beds) and that a physician and a nurse are assigned at each room. This is a simplifying assumption, but actually this is the real situation often encountered in an ED.
Under this assumption, the MIU optimal resource allocation problem consists in deciding the number of rooms to be open and their working hours in each day of the period D. Note that, due to changes in patient flow, the number of MIU rooms to open, as well as opening and closing times, may be different for each day.

5.3.1 The decision variables

To formulate the problem we introduce the following decision variables:

- let $x_d \in \mathbb{Z}^+$ be the *opening time* of MIU on day $d \in D$;
- let $y_d \in \mathbb{Z}^+$ be the *closing time* of MIU on day $d \in D$;
- let $r_d \in \mathbb{Z}^+$ be the *number of rooms* used in MIU on day $d \in D$;

Therefore, we have three n-dimensional integer valued vector variables, namely

$$\mathbf{x} = (x_1, \dots, x_n)^\top, \quad \mathbf{y} = (y_1, \dots, y_n)^\top, \quad \mathbf{r} = (r_1, \dots, r_n)^\top. \tag{5.1}$$

Therefore, $(\mathbf{x}, \mathbf{y}, \mathbf{r}) \in \mathbb{Z}^n \times \mathbb{Z}^n \times \mathbb{Z}^n$ represent a MIU setting for the time period D. Note that variables x_d and y_d are assumed to be integer since usually opening and closing hours refer to exact hours (e.g., 8.00 a.m.–8.00 p.m.)

5.3.2 The sample response function and the sample average approximation

Given a time period D, for each tag $t \in \mathcal{T}$ and for each $u \in U(t)$, we define *sample response function* the function

$$SRF_{t\,u}(\mathbf{x}, \mathbf{y}, \mathbf{r}; \xi(\omega))$$

which takes two inputs: the MIU setting $(\mathbf{x}, \mathbf{y}, \mathbf{r})$ and $\xi(\omega)$, being the latter a random vector defined on a probability space. More precisely, the random realizations of $\xi(\omega)$ correspond to different flow patients through the ED. Sample response functions are

often related to waiting times or stay time in the ED, like DTDT or LOS, or they can be defined in correspondence with some counters, like number of patients in the ED or waiting in a queue.
Sample average approximation method, also called *sample path method* (see, e.g., [72],[74], [108], [112]) consists in approximating the expected values (mathematical expectation) of a sample response function with deterministic averages. By using this approach, the expected value $\mathbb{E}\left[SRF_{t\,u}(\mathbf{x},\mathbf{y},\mathbf{r};\xi(\omega))\right]$ is approximated by a deterministic average sample response function, namely

$$\mathbb{E}\left[SRF_{t\,u}(\mathbf{x},\mathbf{y},\mathbf{r};\xi(\omega))\right] \approx \frac{1}{N}\sum_{i=1}^{N} SRF_{t\,u}(\mathbf{x},\mathbf{y},\mathbf{r};\xi_i), \tag{5.2}$$

where ξ_i is a realization of $\xi(\omega)$ and N is the number of samples. Since a DES model is adopted for generating such realizations, actually, each sample corresponds to the output of a single replication of the simulation runs and the r.h.s. of (5.2) is computed as average over N independent replications.

5.3.3 The objective functions

Now we define the functions of the MIU optimal resource allocation problem. The first one considers the average of the waiting times DTDT for all the patients over the time period D, namely

$$f_1(\mathbf{x},\mathbf{y},\mathbf{r}) = \sum_{t\in\mathcal{T}} \alpha_t \sum_{u\in U(t)} \beta_u\, \mathbb{E}\left[DTDT_{t\,u}(\mathbf{x},\mathbf{y},\mathbf{r};\xi(\omega))\right], \tag{5.3}$$

where $\alpha_t > 0$ and $\beta_u > 0$ are suited scalars and $DTDT_{t\,u}(\mathbf{x},\mathbf{y},\mathbf{r};\xi(\omega))$ denotes the door-to-doctor time of a t-tagged patient assigned to the ED unit u when the ED setting during the period D is given by $(\mathbf{x},\mathbf{y},\mathbf{r})$. Therefore, the real valued function f_1 is computed by taking an expectation over the sample response function and hence, even if the sample average approximation is used, it has no explicit form.
The second one counts the MIU working hours over the time period D, namely

$$f_2(\mathbf{x},\mathbf{y}) = \sum_{d\in D} \gamma_d(y_d - x_d) \tag{5.4}$$

where $\gamma_d > 0$ are suited scalars.
Finally, we consider the number of MIU rooms which are open for each day over the time period D,

$$f_3(\mathbf{r}) = \sum_{d\in D} \delta_d r_d, \tag{5.5}$$

where $\delta_d > 0$ are suited scalars.
Note that all the scalars introduced (5.3), (5.4), and (5.5) aim at possibly differently weighting the single terms of the functions. Moreover, observe that each MIU working hour implies operating costs as well as the more MIU rooms open, the higher the costs. Therefore, since the goal is that of minimizing the average patient waiting

time along with MIU operating costs, we formulate the optimization problem as the simultaneous minimization of function f_1, f_2 and f_3, thus leading to a MO problem with conflicting objectives (see Sect. 5.3.5).

5.3.4 The constraints

Simple box constraints on the variables must be introduced, namely

$$\begin{aligned} l_{x_d} \leq x_d \leq u_{x_d} \\ l_{y_d} \leq y_d \leq u_{y_d} \\ l_{r_d} \leq r_d \leq u_{r_d}, \end{aligned} \tag{5.6}$$

for all $d \in D$, where l_{x_d}, l_{y_d}, l_{r_d} and u_{x_d}, u_{y_d}, $u_{r_d}$are non–negative integer prefixed lower and upper bounds, respectively. Moreover, the constraints

$$y_d - x_d \geq h_d \tag{5.7}$$

for all $d \in D$ must be considered, where $h_d \geq 0$ is a prefixed integer lower bound on the minimum number of hours for MIU to be open. Indeed, it could be necessary to impose that MIU is open on some day week for a number of hours greater than a suited threshold value.
Another constraint regards a lower bound on the overall number of MIU working hours in the time period D, namely

$$\sum_{d \in D} (y_d - x_d) \geq g, \tag{5.8}$$

where g is a prefixed non–negative integer. This is necessary to guarantee that MIU is open for an appropriate time during the whole period D.
We denote by $\mathcal{F}$ the feasible set, i.e., the set of points satisfying all the constraints.

5.3.5 The multiobjective Simulation–Based Optimization problem

The MIU optimal resource allocation problem we are dealing with can be stated as

$$\min_{(\mathbf{x},\mathbf{y},\mathbf{r}) \in \mathcal{F}} \left(f_1(\mathbf{x},\mathbf{y},\mathbf{r}), f_2(\mathbf{x},\mathbf{y}), f_3(\mathbf{r})\right)^\top, \tag{5.9}$$

i.e., as a multiobjective integer optimization problem. The aim is to find a trade–off between the conflicting objectives of reducing the management cost and guaranteeing patients timely treatments according to their urgency code.
A major issue must be taken into account when tackling problem (5.9). In fact, the complex and stochastic processes within the ED prevent us from using analytical models to represent the ED operation and, as we already observed, simulation models are commonly adopted to this aim. In particular, we refer to DES models which proved to be a flexible and robust tool for modeling patient flow through an ED. By using this approach, after building and validating an ED DES model, the patient waiting time sample response function is evaluated via simulation runs. Therefore, the problem we are considering is actually a *Mutiobjective Integer Simulation–Based Optimization* problem.

5.4 Methodology

In this section we describe the methodologies we are going to use for the solution of the SBO problem (5.9). In particular, we propose:

- the use of Derivative-Free optimization algorithms within a method for multi-objective optimization;
- a novel methodology based on the use of Artificial Neural Networks.

A description of both the methodologies is reported in the following two subsections.

5.4.1 Solution of the multiobjective SBO problem via Derivative–Free methods

For an SBO problem, the objective and constraint functions have no analytic form and are subject to the random random noise in their evaluation. This leads to the impracticability of using optimization algorithms based on first–order information (or their approximation). In particular, in problem (5.9), objective function f_1 is evaluated by running a DES model. Therefore, a methodology suited for solving this class of problems is Black–Box and Derivative–Free Optimization (DFO) (see, e.g., [30], [10]). In particular, we focus on direct–search methods of directional type and, to the best of our knowledge, this is the first time that a globally convergent DFO algorithm is used to solve a resource allocation problem in an ED.
Moreover, problem (5.9) is also a multiobjective problem, hence requiring suited solution algorithms (see, e.g. [11, 81, 89]). We briefly recall that MO methods are usually classified on the basis of the articulation of the decision maker preferences[2]. More precisely, we have: *i)* methods with *a priori articulation of preferences*, requiring that the decision maker has to state additional preferences, prior to the optimization, and a single optimal point is then returned as a result of such preferences; *ii)* methods with *a posteriori articulation of preferences*, requiring that the decision maker provides the preferences a posteriori, that is after being informed about non–dominated solutions.
In the first case, objective functions are combined into a single one according to a suited aggregation criterion before the optimization procedure starts (e.g., weighted sum method), hence, a single non–dominated point is obtained. Instead, methods belonging to the second class try to obtain the Pareto front of the MO problem, giving the decision maker the choice of selecting the best point according to his/her preferences We also point out that this is a common classification also in the DFO literature [29, 35, 82].
We adopt the first approach recalled above to deal with the presence of multiple objectives in (5.9) and we focus on the use of weighted sum method (scalarization

[2]For the sake of completeness, we note that methods without preferences can be adopted as well [86], but we do not consider them.

technique) [44, 89]. Namely the objective functions are weighted into a single objective, i.e., the following function

$$F(\mathbf{x}, \mathbf{y}, \mathbf{r}) = \eta_1 f_1(\mathbf{x}, \mathbf{y}, \mathbf{r}) + \eta_2 f(\mathbf{x}, \mathbf{y}) + \eta_3 f_3(\mathbf{r}), \tag{5.10}$$

is defined, with $\eta_1 \geq 0$, $\eta_2 \geq 0$, $\eta_3 \geq 0$ and $\eta_1 + \eta_2 + \eta_3 = 1$. Then several functions F obtained by considering different combinations of the weights η_1, η_2, η_3 are minimized, obtaining an approximate Pareto front [11]. Therefore, although scalarization techniques are originally included among the methods using *a priori* articulation of preferences, the approach we adopt is actually an *a posteriori* strategy since it allows recovering a subset of points in the Pareto front. To perform these several single objective minimizations, we adopt Algorithm DFLINT proposed in [83], which has been proved to have, under some assumptions, finite convergence to a suitably defined local minimum, and to be very efficient for solving integer black–box constrained problems. This is due to the use of particular search directions and a suitable nonmonotone linesearch, guaranteeing a high level of freedom to cover all the feasible points.
As far as the authors are aware, the adoption of such a globally convergent DFO methods represents an important novelty in tackling optimization problems arising in ED management contexts, where heuristics are the most commonly adopted approaches.

5.4.2 Solution of the multiobjective SBO problem via Artificial Neural Networks

The fact that, in the SBO approach, runs of a DES model are needed to evaluate objective and constraint functions implies that the overall optimization procedure is very time consuming. Of course, the required computational effort depends on the complexity of the simulation model and on the length and number of replications of each run. However, to ensure a good accuracy of the results, usually such runs are very time expensive. This leads to a long CPU time required to solve SBO problems by means of approaches based on a direct use of a DES model.
To overcome this drawback, we introduce the use of an ANN in order to efficiently determine a subset of points in the Pareto front of the multiobjective SBO problem under study. The key idea is to construct a machine learning model that "emulates" the DES model enabling the computation of the value of the objective function (5.3) in a significantly cheaper way, albeit less precise. On the other hand, if we assume that a fixed "budget" of function evaluations is available, by using the approaches described in Sect. 5.4.1, the quality of the obtained solution strictly depends on this budget. In particular, in the scalarization approach, such budget would be distributed in the solution of the several problems corresponding to different choices of the weights; thus, as the number of these problems grows, the budget for each problem and, consequently, the quality of their solution decreases significantly.

As well known and already discussed in Sect. 3.2 ANNs are computing systems that are inspired by the biological neural networks that compose animal brains (for a detailed description of ANNs we refer, e.g., to [48, Chapter 11] and [61, Chapter 2]). These computing systems have the ability to capture the complex interactions among inputs via their hidden neurons. The neurons are units of computation that receive input from other neurons, make computations on these inputs (such as performing a weighted sum before applying a nonlinear function, commonly known as activation function), and feed them into other neurons. In Sect. 3.2 we showed that, if the ANN has the structure of a Multi-Layer Perceptron (MLP), the network is known to have universal approximation properties. Our belief is that a sufficiently precise machine learning model that captures the essential behavior and relationships between the input and the output variable can be efficiently used as alternative to a DES model. Moreover, this approach also allows the solution of the SBO problem by means of gradient–based methods, since, in this case, an analytic expression of the objective function is available. The complexity of this function depends on the number of hidden layers, the number of neurons and the activation functions of the MLP, but, regardless of the complexity, it will always be possible to compute its gradient.
We now describe the ANN–based approach we propose for tackling the SBO problem (5.9). It is characterized by the following sequential steps:

1. *generation of the dataset* that will be used by the MLP;
2. *identification of good hyperparameters* for the training problem;
3. *training* of the MPL;
4. *generation of a Pareto front* for the SBO problem (5.9) using the MPL for evaluating the objective function f_1 in (5.3).

In the following subsections we provide a detailed description of these steps.

Generation of the dataset

To train the model under the supervised learning paradigm, a dataset with the following structure is needed: $\{\mathbf{v}_p, w_p\}_{p=1}^{P}$ where $\mathbf{v}_p$ represents the input vector and w_p a scalar representing the target variable. For the problem under investigation, each vector $\mathbf{v}_p$ consists in the concatenation of the n-dimensional integer vectors $\mathbf{x}$, $\mathbf{y}$ and $\mathbf{r}$ defined in (5.1). Therefore, the vector $\mathbf{v}_p$ is a $(3n)$-dimensional vector

$$\mathbf{v}_p = (x_1, \dots, x_n, y_1, \dots, y_n, r_1, \dots, r_n)^\top \tag{5.11}$$

representing the ED setting during the whole period $D = \{1, \dots, n\}$.
After creating P random vectors $\mathbf{v}_1, \dots, \mathbf{v}_P$ satisfying constraints (5.6), (5.7) and (5.8), the associated target values $w_1, \dots, w_P$, necessary for the training of the MLP, are obtained by means of runs of the DES model. In particular, from the simulation output we extract the sample average approximation of the time-to-doctor time

DTDT corresponding to each ED setting $\mathbf{v}_p$ and store it in the scalar w_p. It is worth highlighting that by generating the dataset in this way, we do not encounter many of the issues commonly faced when dealing with ML models, like the presence of missing values, the presence of outliers and the fact that sometimes samples do not come from the same probability distribution. In fact, the use of runs of the DES model for generating data ensures high quality data, enabling the control of the size of the dataset (that can be adjusted depending on the performance of the model), and helping train a reliable ML model. Note that, due to the presence of high degree of nonlinearity and complex relationships between the vectors $\mathbf{v}_p$ and the target variables w_p, a MLP with a sufficiently large number of hidden layers and neurons is usually identified as a good ANN model to learn this relationship [61].
It is important to observe that the procedure now described for the generation of the dataset is the only time–consuming phase of the ANN approach we propose and it can be easily parallelized.

Identification of the hyperparameters

Before training a MLP, it is necessary to decide the values of the hyperparameters involved in the training process. They consist of parameters that define the architecture of the network, like the number of layers, the number of neurons of each layer, the dropout rate, and of the parameters that play a role in the training process, like the parameters of the optimizing algorithm, the number of epochs and the batch size. For the importance of the hyperparameters in the model's performance and for an overview of different selection strategies we refer to [28].
To choose a good configuration of the hyperparameters we implemented a random search with a K–fold Cross-Validation, a resampling procedure commonly used to choose the values of the hyperparameters without affecting the assessment of the generalization capabilities of the model [19]. It works as follows: the training set is split into K groups (folds). Then, K iterations are performed and, at each iteration $k = 1, \dots K$, $K-1$ folds are used as training sets excluding the k-th one and using this latter as validation set. The network is trained with a given configuration of the hyperparameters and an evaluation metric (e.g., the mean absolute error) is computed on the validation set. After the last iteration, the average values of the metric across the K different validation sets are computed and stored. Then, the configuration of the hyperparameters is chosen as the one which minimizes the average values of the chosen metric.

Training of the MLP

Training a machine learning model consists in finding the values of all the model parameters so that the model error is minimized, in other words, the distance between the value of a certain indicator of interest predicted by the model and the corresponding real value is small. The training procedure is performed by means of

the solution of an optimization problem using Stochastic Gradient Descent algorithm [22]. Once the model is trained, we then have an analytical formula for computing the value of function f_1 in (5.3) of the MIU optimal resource allocation problem (5.9).

Generation of the Pareto frontier

We can obtain a Pareto front or a subset of non–dominated points approximating the front by using the weighted sum methods tacking into account that the function f_1 is now replaced by its approximation obtained via the machine learning model. In this regard, some major important issues are worth highlighting:

- the evaluation of the objective function f_1 is extremely faster than its simulation–based alternative;
- unlike the approach based on DFO methods, since the function f_1 is replaced by its approximated model, all the problems functions are now continuously differentiable. This enables using gradient–based optimization methods and hence a great efficiency improvement is achieved;
- parallel or distributed computing techniques can be used. In fact, the use of concurrent processes can be adopted for generating the dataset, and hence different minimizations can be performed simultaneously;
- if needed, further constraints which do not involve KPIs computed via the simulation model can be easily added to the initial problem formulation without re–building the model from scratch.

Therefore, following the approach based on ANN we propose, once the data set has been generated, each single objective minimizations can be performed in a really short computational time. Thus, a great number of such minimizations can be carried out, leading to the generation of more points in the approximate Pareto front.
Note that, an additional (optional) final step of the procedure can be considered. In fact, the value of the objective function f_1 corresponding to each point in the Pareto front actually is an approximation of the expected value of the door-to-doctor DTDT patient waiting time. Therefore, in order to have a greater accuracy of this value, each obtained point of the Pareto front can be selected as input of the DES model and the DTDT computed via a simulation run. We will adopt this additional step in our computational experiences reported in Sect. 5.7, in order to perform a fair comparison between the results obtained via the ANN approach and those resulting from the use of a DFO algorithm within the weighted sum method.

5.5 The case study

In order to experiment the approaches we propose in this paper, and in particular

the one based on ANN described in Sect. 5.4.2, we consider a real case study, namely the ED of a large Italian hospital which is significantly affected by the problem of overcrowding: the ED of *Policlinico Umberto I* in Rome, Italy, one of the largest EDs in Europe with about 140,000 patients arriving each year. It is composed of several areas addressing medical specialties that range from ophthalmology and hematology to obstetrics, pediatrics, and dentistry. In this paper, we focus on the *central area*, which provides treatments to patients suffering from diseases and disorders within the scope of internal medicine and general surgery. The real data used in our computational experiments was collected during 2018, when more than 50,000 patients used the emergency services offered by the central area of this ED. Note that, in 2018, a four–level (color–based) triage scale were used, namely *red*, *yellow*, *green* and *white* tags (listed in decreasing order of severity) were assigned[3]. Some important aspects of this case study have been already deeply analyzed in the papers [41] and [40]. In particular, in [41] patient arrivals process is examined in depth in order to determine the optimal piecewise constant approximation for the nonhomogeneous Poisson process arrival rate. Paper [40] deals with the problem of missing data, which affects many processes within the ED; in particular, a model calibration procedure is proposed to obtain a DES model which produces an accurate output. We refer to these two papers for a detailed description of the ED operation. In the sequel, for the sake of completeness, we briefly report those aspects that are more relevant with respect to the problem we are dealing with.

5.5.1 The ED units

Two main units are used to visit and treat the patients from all the triage categories: a *Medical Unit* (MU), which is specific for diseases and disorders related to internal medicine, and a *Surgical Unit* (SU), which provides treatments to patients needing a surgical operation. Within these units, fully–equipped areas, denoted as MU_{Red} and SU_{Red}, are dedicated to red–tagged patients. Moreover, a *Resuscitation Area* (RA) is available when the health conditions require increasing level of attention due to life–threatening diseases. In contrast, yellow–tagged and green–tagged patients do not have dedicated rooms and share the same resources within MU and SU. Finally, a *Minor Injuries Unit* (MIU) is devoted to the visit and treatment of the least urgent patients, namely, the least critical green–tagged patients and all the white–tagged patients which are directed to the MIU by the triage nurse via a FT. Usually, MIU is open Monday through Saturday 8:00 a.m.–8:00 p.m., and green tag is assigned in place of the white one when MIU is closed. All the other units are open 24 hours a day, 7 days a week. As regards the number of rooms used for the medical visit and treatment of red–tagged patients, RA has 2 rooms always available, while MU and SU have 1 and 2 rooms, respectively. With regard to the other patients, the rooms used in MU and SU are 3 and 2 in the day time (8:00 a.m.–8:00 p.m.) and 2 and 1 at night (8:00 p.m.–8:00 a.m.), respectively. Finally, 2 rooms are typically used in

[3]More recently, a five–level based scale has been adopted

MIU.

5.5.2 The patient flow

Several diagnostic and therapeutic pathways can be identified within the ED on the basis of the severity tag assigned to patients. In Fig. 5.1 we report a simplified scheme of the patient flow, highlighting the major blocks. In particular, in the ED

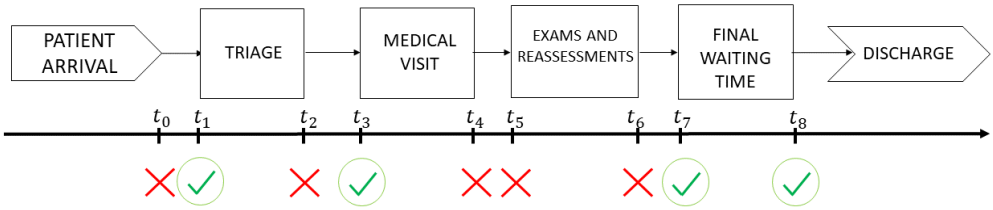

Figure 5.1. Scheme of ED patient flow and related timestamps. A green tick is used to indicate that a timestamp is available from data.

under study red–tagged patients can be directed to MU_{Red} or SU_{Red} on the basis of the medical specialty required, or to RA in case of critical health conditions. Yellow–tagged patients are visited and treated only in MU and SU, where resources are shared with green–tagged patients that are not sent to MIU. The latter decision, which is made by the nurse in charge of triage, depends on two conditions: on the one hand, whether MIU is open or closed; on the other hand, the severity of the patient's disease. In the MIU, the green–tagged patients share the available resources with the white–tagged patients. After the medical visit and treatment, the subsequent stage is composed of exams (CT scans, X-rays, and so forth) and reassessments. Some patients may be also required observation periods among the different exams or before the discharge. A complete description of all the clinical pathways adopted in this ED can be found in [40].

5.5.3 Data collection

Data collection represents a key point in order to develop an accurate ED model. For the purpose of this study, we consider the data collected in January, since this month is considered to be particularly critical in terms of overcrowding level by the ED managers (of course, different months or time periods can be easily considered). In January 2018, 4192 patients arrived in the central area of the ED of Policlinico Umberto I. Table 5.1 reports their distribution among severity tags and ED units assigned, excluding LWBS and deceased patients, and those directed to the orthopedic unit which is not considered in this study, since it concerns a specific dedicated pathway, not involving MIU. The table clearly evidences that the majority of patients are yellow-tagged. Moreover, most of these patients are directed to MIU, highlighting a sharp difference when compared with green-tagged patients, which

Table 5.1. Distribution of patients among ED units and color tags

	White	Green	Yellow	Red	
MU	-	248 (21.83 %)	1316 (65.51 %)	191 (75.79 %)	1755
SU	-	628 (55.28 %)	693 (34.49 %)	45 (17.86 %)	1366
RA	-	-	-	16 (6.35 %)	16
MIU	47 (100 %)	260 (22.89%)	-	-	307
	47 (100 %)	1136 (100 %)	2009 (100 %)	252 (100 %)	

are mostly sent to SU. Finally, most of the red-tagged patients require visit and treatment in MU, thus showing a similar behaviour as yellow-tagged ones.
While data concerning tags assigned and resources scheduled are usually sufficiently complete (assigned tags, physician and nurse shifts, room available etc. are usually recorded), data regarding the timestamps of each process inside an ED are often lacking. This is a well known problem in the literature on EDs (see [40] and the references reported therein). As regards our case study, Fig. 5.1 reports the timestamps of the main processes in the patient flow, highlighting those available and unavailable. For a detailed description of all the timestamps and the motivations of unavailability of some of them, we refer to the paper [40]. We just recall here that, since timestamps t_2, t_4, t_5, and t_6 are unavailable, we cannot directly compute the service times of triage, medical visit, and examinations and reassessments from the data and we use the procedure described in [40] to estimate these service times.

5.5.4 The Discrete Event Simulation model

A DES model of the patient flow within the ED has been implemented by using the R package Simmer [128], which allows building process-oriented DES models based on trajectories. For a complete description of the DES model of the ED under study, we refer to paper [40]. Here we just recall that the model entities are patients which enter the model according to a proper arrival process, namely a piecewise constant approximation of a nonhomogeneous Poisson process (see [41] for a detailed description of this process).
After the triage, entities follow different trajectories according to the severity tag received (which is stored as an attribute) and to the ED unit assigned. The combination given by the color tag and the ED unit determines the different patient flows.
The resources used in the model represent the rooms for the medical visit and treatment. While the room of MU_{Red} SU_{Red} and RA have a fixed capacity, the rooms of MU, SU and MIU are based on a schedule that reflects the change in the capacity between the day and the night shifts.
For the model validation and the design of experiments we refer to the paper [40] and we adopt here the same choices, namely 38 days as simulation length, 1 week as

warm up period and the number of replications fixed to 30.

5.6 Statement of the problem for the case study

The simulation model of the ED of Policlinico Umberto I is used to achieve the final goal of finding a setting that allows reducing the level of overcrowding, which is measured through the waiting times at each ED unit. By interviewing the ED managers, a great interest emerged about exploring the existence of a margin for visiting and treating in MIU a significantly larger number of green-tagged patients. This request is motivated by the fact that triage nurses in charge of assigning the color tag at triage usually tend to assume a precautionary behavior, thus sending fewer patients to MIU (via FT), than its actual capacity. Indeed, they would be deemed responsible in case of worsening of the patient health conditions caused by a wrong choice of the ED unit at triage. However, reducing the overcrowding by leveling out the workload within the ED units requires to carefully monitor the overall number of patients to avoid that the benefit achieved in one unit gives rise to long waiting times in other segments of the ED. Sending more patients to MIU is only one of the options available to achieve the final goal of reducing the overcrowding. A further tool considered by the ED managers is a proper choice of the working hours of MIU, which is currently open from 8:00 a.m. to 8:00 p.m., Monday through Saturday. Due to the high flexibility of the ED staff, every combination of opening and closing time for MIU on each day of the week is considered feasible by the managers if an improvement in the current status is guaranteed, provided that the working hours are not out of the time window 7:00 a.m.–8:00 p.m. (only an early opening of one hour is possible with respect to the current setting). Of course, the largest reduction in the waiting times is expected for low-complexity patients, namely patients with green or white triage tag, since these are the categories assigned to MIU. As observed in [121], more critical patients may experience a lower reduction in the waiting times since their priority guarantees a shorter waiting time. In particular, some benefits are expected for yellow-tagged patients, who will share resources with a lower number of low-complexity patients. In contrast, red-tagged patients rely on dedicated resources and, accordingly, no improvement is expected for them.
Now let us formally state the optimal resource allocation problem for MIU of the ED of Policlinico Umberto I. By using the notation introduced in Sect. 5.3, we have $\mathcal{T} = \{R, Y, G, W\}$, the set of severity (color) tags, red (R), yellow (Y), green (G) and white (W). Moreover for each $t \in \mathcal{T}$, $U(t) \subseteq \{\mathrm{MU}, \mathrm{SU}, \mathrm{RA}, \mathrm{MIU}\}$, namely

$$U(t) = \begin{cases} \{\mathrm{MIU}\} & \text{if} \quad t = W, \\ \{\mathrm{MU}, \mathrm{SU}, \mathrm{MIU}\} & \text{if} \quad t = G, \\ \{\mathrm{MU}, \mathrm{SU}\} & \text{if} \quad t = Y, \\ \{\mathrm{MU}, \mathrm{SU}, \mathrm{RA}\} & \text{if} \quad t = R. \end{cases}$$

Since a weekly planning is adopted, we have $D = \{1, \ldots, 7\}$. The decision variables

are opening and closing hours for each week day, namely x_d and y_d, $d \in D$. The variables r_d and the objective function f_3 in (5.5) are not introduced, since in the ED under examination, the number of rooms used in MIU is prefixed. The resulting SBO problem for the optimal resources allocation for MIU of the ED under study is then formulated as

$$\min_{(\mathbf{x},\mathbf{y})\in\mathcal{F}} (f_1(\mathbf{x},\mathbf{y}), f_2(\mathbf{x},\mathbf{y}))^\top, \tag{5.12}$$

where the feasible set $\mathcal{F}$ is defined by the constraints (5.6), (5.7) and (5.8) and the function f_1 and f_2 are defined in (5.3) and (5.4), respectively. It is a simulation–based bi–objective integer optimization problem, with 14 variables and 21 constraints. The sample average approximation method is used, as described in Sect. 5.3.5, to handle the expected value in function f_1. In particular, runs of the ED DES model recalled in Sect. 5.5.4 are used to computed the average sample response function of the waiting time of interest, namely the door-to-doctor time DTDT given by $t_3 - t_0$ in Fig. 5.1. Actually, we approximate such time, with the time difference between the start of the triage and the start of the medical visit, namely $t_3 - t_1$. This is due to the fact that, as we already noticed, arrival time t_0 is unavailable from data, hence it can not be considered in the implementation of the ED DES model. On the other hand, by assuming that (as patient would expect) the triage starts when a patient arrives in the ED (i.e., $t_0 = t_1$), the difference between DTDT and the time difference $t_3 - t_1$ can be considered negligible.

5.7 Experimental results

In this section we report the results of an experimentation of the approaches we propose for solving the MIU optimal resource allocation problem arising in the case study described in Sect. 5.5. In this experimentation, we consider the problem (5.12) where the function f_1 is defined in (5.3) with $\alpha_t = 1$ for all $t \in \mathcal{T}$ and $\beta_u = 1$ for all $u \in U(t)$; the function f_2 is defined in (5.4) with $\gamma_d = 1$, for $d = 1, \ldots, 7$. This choice is motivated by the will of not differently weighting the terms that make up the functions. As regards the problem constraints, in the box constraints (5.6), lower and upper bounds on the variables x_d and y_d are set to 7 (7:00 a.m.) and 20 (8:00 p.m.), respectively, namely,

$$l_{x_d} = l_{y_d} = 7, \qquad u_{x_d} = u_{y_d} = 20, \qquad d = 1, \ldots, 7.$$

In the constraints (5.7) we set $h_d = 0$ for $d = 1, \ldots, 7$, to allow possible closing of the MIU for some day week. In (5.8) we choose $g = 21$ to ensure a certain number of MIU open hours during the week, as suggested by ED managers.
In the sequel, we report the results obtained by applying both the methodologies we propose, as described in Sect. 5.4. All the tests were performed on a PC with an Intel Core i7-4790K Quad–Core 4.00 GHz Processor with 32 GB RAM. We assume that the maximum number of function evaluations allowed is set to $10,000$.

5.7.1 Results from the application of DFO methods

We consider the weighted sum method and an approximate Pareto front for problem (5.12) is computed by means of several minimizations of the function $\eta_1 f_1(\mathbf{x}, \mathbf{y}) + \eta_2 f_2(\mathbf{x}, \mathbf{y})$ over the feasible set $\mathcal{F}$, considering different combinations of the weights $\eta_1 \geq 0$, $\eta_2 \geq 0$, with $\eta_1 + \eta_2 = 1$. In particular, we consider the following 11 combinations of the weights (η_1, η_2): $\{(1,0),\ (0.9,0.1),\ (0.8,0.2),\ \cdots\ ,(0.1,0.9),\ (0,1)\}$. Each single objective minimization is performed by means of the DFLINT algorithm [83]. This algorithm is available with GNU GPL license on the web[4]. The starting point of each minimization corresponds to the "as–is" status, namely to the current MIU opening and closing times $\mathbf{x}^0$ and $\mathbf{y}^0$. The value of each component of this starting point is reported in Table 5.2 (note that MIU is currently closed on Sunday). From a run of the DES model we obtain $f_1(\mathbf{x}^0, \mathbf{y}^0) = 724,04$ minutes and a simple

x_1^0	x_2^0	x_3^0	x_4^0	x_5^0	x_6^0	x_7^0	y_1^0	y_2^0	y_3^0	y_4^0	y_5^0	y_6^0	y_7^0
8	8	8	8	8	8	8	20	20	20	20	20	20	8

Table 5.2. Starting point of the optimization algorithms corresponding to the "as–is" status

computation gives $f_2(\mathbf{x}^0, \mathbf{y}^0) = 72$ hours (see the green square in Fig. 5.2).
As regards the parameters of the DFLINT algorithm, the default ones are adopted. The stopping criterion is based on the maximum number of function evaluations, so that the overall prefixed "budget" of 10,000 function evaluations is equally divided into the 11 applications of the DFLINT algorithm, leading to the stopping criterion of 909 function evaluations allowed for each minimization. The values of functions f_1 and f_2 corresponding to the optimal point of the single objective minimizations are reported in Table 5.3, removing those corresponding to coinciding and dominated points. The approximate Pareto front consisting of such points is reported in Fig. 5.2

	$f_1^\star$	$f_2^\star$
A	714.84	58
B	721.52	54
C	724.19	51
D	755.73	40
E	785.49	23
F	786.83	21

Table 5.3. Results for the weighted sum method: optimal values obtained in the single objective minimizations ($f_1^\star$ in minutes, $f_2^\star$ in hours)

with blue triangle.
Results in Table 5.3 clearly highlight the fact that, as expected, high values of the weight η_2 leads to a setting for which opening hours during the week are very reduced,

[4]See the DFL Library at `http://www.iasi.cnr.it/~liuzzi/DFL`

with a corresponding increase of the waiting times. It also worth noting that, in general, to the same value of weekly opening hours may correspond different settings and, as a consequence, different waiting times. For the same reason, larger value of opening hours not necessarily imply a decrease of the waiting times. The ED setting corresponding to these solution points, are reported in Table 5.4. Of course, some

	$x_1^\star$	$x_2^\star$	$x_3^\star$	$x_4^\star$	$x_5^\star$	$x_6^\star$	$x_7^\star$	$y_1^\star$	$y_2^\star$	$y_3^\star$	$y_4^\star$	$y_5^\star$	$y_6^\star$	$y_7^\star$	
A	9	9	7	9	10	9	9	15	17	19	20	20	16	13	30
B	10	8	8	9	10	8	9	18	17	19	16	16	19	11	
C	8	7	7	11	7	10	10	13	15	15	14	20	15	19	
D	19	9	7	7	7	7	8	20	13	15	7	18	20	11	
E	19	7	9	20	12	8	10	20	10	13	20	15	19	11	
F	19	20	10	20	7	11	20	20	20	19	20	15	14	20	

Table 5.4. Results for the weighted sum method

settings, even if feasible, could be improper for ED managers to adopt in practice. However, once managers are provided with a set of solutions points belonging to the Pareto front, they can choose those more suited to their need and this is a really significant advantage of the approach we propose. As a consequence, it is important to generate a sufficiently large number of these non–dominated points. However, as we already discussed, the application of DFO methods within the SBO approach requires a great computational effort since function evaluations are performed by means of runs of the DES model. This prevents us from considering a greater number of combinations of the weights η_1 and η_2 in the weighted sum method. In fact, to avoid exceeding the prefixed overall maximum number of function evaluations, an increase of the number of minimizations would lead to a decrease of the maximum number of function evaluations allowed for each single objective minimization, and this compromises a good accuracy of the results. This is the rationale behind our choice to also propose an alternative approach based on ANN whose results are reported in the following section.

5.7.2 Results from the application of ANN approach

As already pointed out in Sect. 5.4.2, the only time–consuming phase of the ANN approach is the generation of the dataset. The time needed for the training phase and the resolution of the different optimization problems is in the order of seconds or a few minutes at most, and is negligible compared to time required for the data generation, which takes approximately 240 seconds for each function evaluation. In the dataset generation phase we use the whole "budget" of 10,000 function evaluations performed by means of runs of the DES model. We divide our data into a training set with 80% of the available data and a test set with the remaining 20%. To find the best configuration of hyperparameters we execute a random search with a 5-fold cross validation. After trying 250 different combinations of hyperparameters

the one that leads to the lowest average mean absolute error on the five different folds is the following: 1 hidden layer, 95 neurons, a *ReLU* activation function and a dropout rate of 0 regarding the features of the model; the Adam optimizer ([73]) with a learning rate of 10^{-3} for what concerns the hyperparameters of the optimization phase.
Training the network on the 8000 samples results in a MAE of 4.38 on the training set whereas the MAE computed on the test set is equal to 4.67. Considering that the y values in the dataset range between values of 708.6 and 828.9, this confirms that we did not incur neither in an underfitting nor an overfitting situation.
Once the function f_1 is replaced by the approximate model, we can consider a much bigger number of combinations of the weights η_1 and η_2, and this allows us generating many more non–dominated points approximating the Pareto front. In particular, we consider the 101 combinations of the weights (η_1, η_2): $\{(1,0),\ (0.99,0.01),\ (0.98,0.02),\ \cdots,(0.01,0.99),\ (0,1)\}$. Of course, we are aware that in few cases, the use of very similar weights could lead to the same minimizer.
All the steps of the ANN–based approach have been implemented in Python by using the library PyTorch. We then used a gradient–based method, namely the algorithm SLSQP [75] available in the SciPy library for the single objective minimizations, performing a rounding to the nearest integer optimal point.
The values of functions f_1 and f_2 corresponding to the optimal point of the single objective minimizations are reported in Table 5.5, removing those corresponding to coinciding and dominated points. The approximate Pareto front consisting of such

	$f_1^\star$	$f_2^\star$
A	712,53	91
B	718,44	62
C	721,80	60
D	723,96	59
E	724,08	52
F	726,27	50
G	726,28	47
H	735,66	43
I	766,70	23
L	770,90	22
M	774,55	21

Table 5.5. Results for the ANN approach: optimal values obtained in the single objective minimizations ($f_1^\star$ in minutes, $f_2^\star$ in hours)

points is reported in Fig. 5.2 with red circles. The ED settings corresponding to these solution points, are reported in Table 5.6.
We can see that the approximated Pareto Frontier obtained with the ANN approach has more distinct points than the one obtained with the DFO approach. Furthermore, these points are more evenly distributed, allowing ED managers to have a wider range of options from which they can choose the solutions that are best suited to

	$x_1^\star$	$x_2^\star$	$x_3^\star$	$x_4^\star$	$x_5^\star$	$x_6^\star$	$x_7^\star$	$y_1^\star$	$y_2^\star$	$y_3^\star$	$y_4^\star$	$y_5^\star$	$y_6^\star$	$y_7^\star$
A	7	7	7	7	7	7	7	20	20	20	20	20	20	20
B	7	7	7	8	8	7	11	16	14	19	15	20	17	16
C	9	7	7	9	7	7	10	18	14	18	17	18	18	13
D	9	7	7	8	7	7	10	17	14	18	17	18	17	13
E	8	7	7	9	9	10	9	16	14	14	17	20	17	13
F	9	7	7	7	9	10	10	16	14	14	14	19	16	16
G	9	7	8	8	9	10	10	16	14	14	14	19	16	15
H	9	7	7	8	9	13	10	15	13	14	13	20	16	15
I	11	9	10	10	10	14	19	15	14	14	13	15	15	20
L	11	11	10	9	8	11	19	14	15	13	13	13	13	20
M	10	11	10	12	8	10	19	14	15	13	14	13	12	20

Table 5.6. Results for the ANN approach

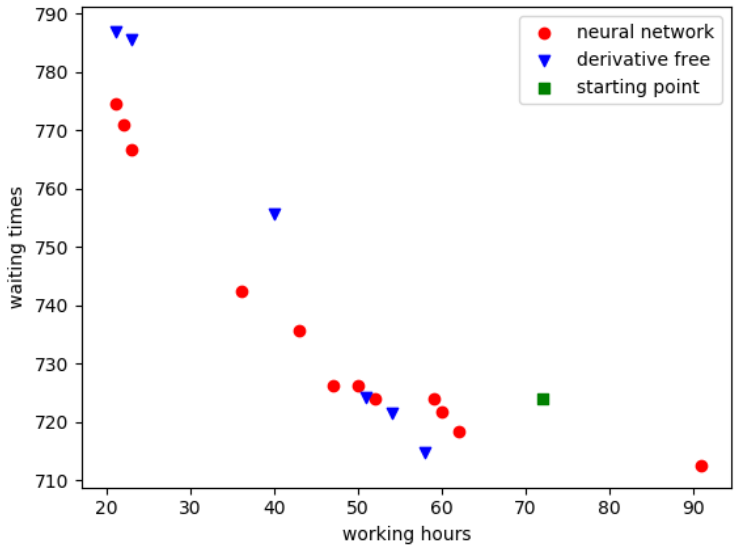

Figure 5.2. Approximated Pareto front obtained by the weighted sum method (blue triangles) and by the use the ANN approach (red bullet). The green square represents the current "as–is" status

their needs.
As for the quality of these points, we can see from Fig. 5.2 that there are three points of the DFO pareto frontier that are dominated by points obtained with the ANN approach, and three points for which the opposite happens. Overall, the Pareto frontier obtained replacing the simulator with the metamodel seems to be better than the other.
Let us also highlight another advantage of the *ANN* approach: the lower computation time. In the DFO approach, for each single objective minimization problem the function evaluations can only be computed sequentially, with each iteration waiting for the previous one to end, without the possibility for them to be parallelized. In the ANN approach, however, there is no constraint on the sequentiality of the generation of the dataset. This means that, the higher the number of processing systems available, the lower the elapsed time required for the computation of the same "budget" of function evaluations. In particular, the *lowest* elapsed time that can be achieved by the DFO approach corresponds to the time necessary for the resolution of one single problem, if all the problems can be solved in parallel. Considering the eleven problems of the DFO approach, the approximate time for the resolution is the time required to perform 909 iterations - i.e. the time necessary for 909 function evaluations.
In the ANN approach, instead, the 10,000 function evaluations can be parallelized on whatever the number of processing systems available is, thus allowing to drastically reduce the elapsed time of the experiment.

5.8 Conclusions

In this chapter, we considered a resource allocation problem related to the ED of Policlinico Umberto I in Rome. The goal of the problem is to determine the optimal settings of the ED unit devoted to the medical visit of low-complexity patients in order to reduce the overcrowding level without an excessive increase in the operating cost. Therefore, a multiobjective formulation of the problem is adopted to find a trade-off between the conflicting goals of reducing the working hours of the ED unit in hand and guaranteeing patients timely treatments according to their urgency code.
Two strategies are adopted to deal with the resulting bi-objective problem, in both of which the two objective functions are weighted into a single function and the resulting optimization problem is repeatedly solved by using a single-objective algorithm in order to determine the approximate Pareto front.
In the first strategy, a globally convergent DFO algorithm is employed. In the second strategy, the usage of a gradient-based algorithm is enabled by replacing the objective function that requires the usage of a simulation software with a metamodel that has the structure of a MLP.
Real data from the patient flow of the Italian hospital are used in the experimental

results. The proposed approaches are able to provide the ED managers with effective tools to determine and assess different settings for reducing the patient waiting times by choosing the desired level of effort. In particular, the ANN approach provides several advantages over the DFO one:

- it enables the resolution of a higher number of single objective optimization problems, thus producing a Pareto Frontier with more distinct points and better distributed;
- it allows to drastically reduce the computation time parallelizing the generation of the dataset on all the processing systems available;
- it is extremely robust to changes in the statement of the problem. If new constraints or objective functions that do not involve the simulation software were to be added to the problem, it would possible to generate a new Pareto frontier rapidly, since it would not be necessary to use the simulation software and the only time necessary would be the one for solving the multiple single-objective minimization using gradient-based methods. The same would also happen if the existing constraints and objective functions were subject to modifications. The DFO approach, instead, requires executing the whole procedure (including the repeated use of the simulation software) even for the slightest change in the statement of the problem.

Future directions of research concern the development of *a posteriori* simulation-based multiobjective algorithms which appropriately tackle the discrete variables without relying on the rounding step and the explorations of different models for approximating the simulation model.

Appendix A

Neural networks activation functions

This Appendix reports some of the activation functions that can be used in neural network models, as discussed in Sect. 3.2.

Among the most popular activation functions we can find:

- the *Heaviside step* function, or *step* function, whose value is zero for negative arguments and one for positive arguments (see Fig. A.1):

$$step(z) = \begin{cases} 1, & \text{if } z \geq 0 \\ 0, & \text{if } z < 0; \end{cases}$$

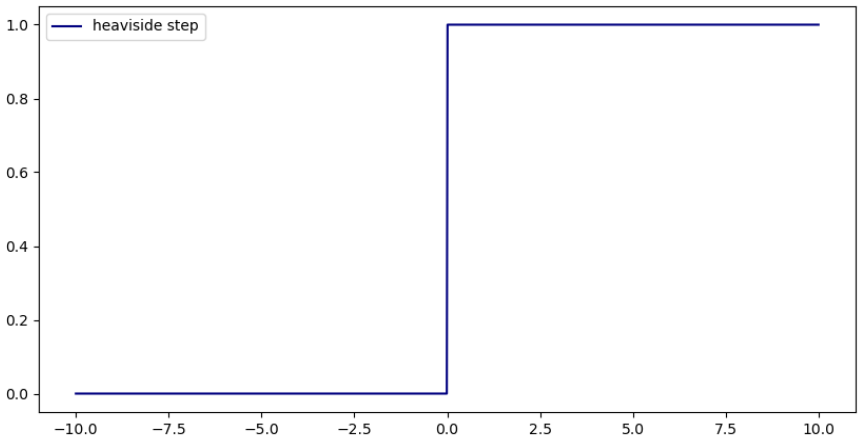

Figure A.1. Heaviside step activation function

- the *sigmoid* function, that prevents the output from being too big by squashing it between 0 and 1 (see Fig. A.2):

$$sigmoid(z) = \frac{1}{1 + e^{-z}};$$

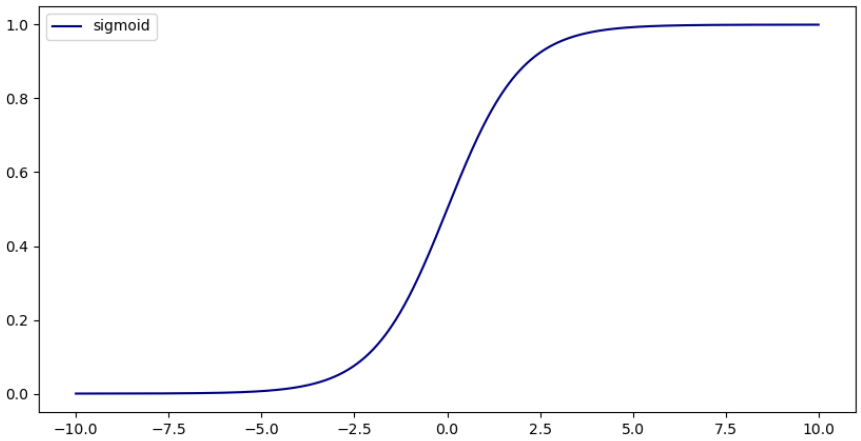

Figure A.2. Sigmoid activation function

- the Hyperbolic Tangent Function, often referred to as *tanh*, that is a smoother zero-centred function whose range lies between -1 and 1 [101] (see Fig. A.3):

$$tanh(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}};$$

- the Rectified Linear Unit ($ReLU$), first proposed in 2010 in [95] and probably the most widely used activation function for deep learning application since (see Fig. A.4):

$$ReLU(z) = max(0, z) = \begin{cases} z, & \text{if } z \geq 0 \\ 0, & \text{if } z < 0. \end{cases}$$

These are only a small subset of the numerous activation functions that have been proposed and used by the Machine Learning community. For more examples and a deeper discussion on the topic we recommend the followings: [69, 101, 110].

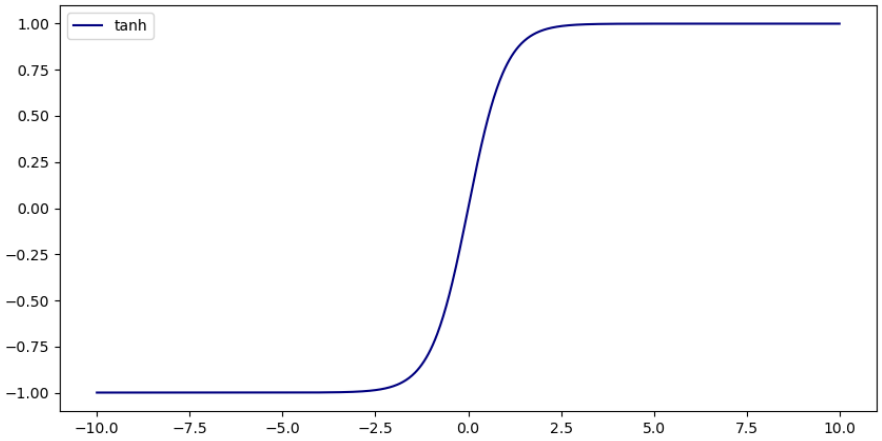

Figure A.3. Hyperbolic tangent activation function

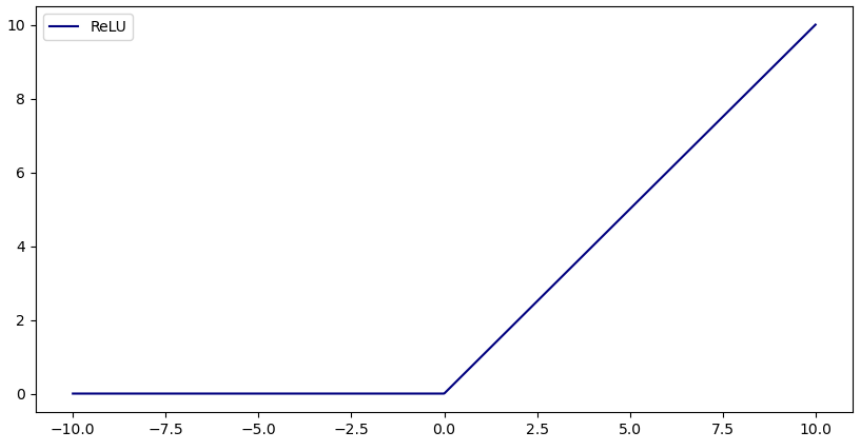

Figure A.4. Rectified Linear Unit activation function

Appendix B

Proof of Theorem 4.1.1

This Appendix we report the Proof of Thm. 4.1.1.

We have that

$$||G_\sigma(x) - \overline{G}_\sigma(x)||^2 = \sum_{i=1}^{n} \Big((G_\sigma(x))_i - (\overline{G}_\sigma(x))_i \Big)^2$$

where $(G_\sigma(x))_i$ is given by (3.26)

$$(G_\sigma(x))_i = \int_{\mathbb{R}^{n-1}} g_\sigma(x_i,\, \bar{x}_i + \sigma \bar{s}_i)\, \varphi(\bar{s}_i)\, d\bar{s}_i$$

and $(\overline{G}_\sigma(x))_i = g_\sigma(x_i,\, \bar{x}_i)$, by (4.1). We can write

$$\begin{aligned} \Big((G_\sigma(x))_i - (\overline{G}_\sigma(x))_i \Big)^2 &= \left(\int_{\mathbb{R}^{n-1}} g_\sigma(x_i,\, \bar{x}_i + \sigma \bar{s}_i)\, \varphi(\bar{s}_i)\, d\bar{s}_i - g_\sigma(x_i,\, \bar{x}_i) \right)^2 \\ &= \left(\int_{\mathbb{R}^{n-1}} \left(g_\sigma(x_i,\, \bar{x}_i + \sigma \bar{s}_i) - g_\sigma(x_i,\, \bar{x}_i) \right) \varphi(\bar{s}_i)\, d\bar{s}_i \right)^2 \end{aligned} \tag{B.1}$$

where the last equality holds since $\int_{\mathbb{R}^{n-1}} \varphi(\bar{s}_i)\, d\bar{s}_i = 1$. Now, the integrand in (B.1) has the following expression

$$\begin{aligned} &g_\sigma(x_i,\, \bar{x}_i + \sigma \bar{s}_i) - g_\sigma(x_i,\, \bar{x}_i) = \\ &\frac{1}{\sigma} \int_{-\infty}^{\infty} \left(f(x_i + \sigma s_i,\, \bar{x}_i + \sigma \bar{s}_i) - f(x_i + \sigma s_i,\, \bar{x}_i) \right)\, s_i\, \varphi(s_i)\, ds_i, \end{aligned} \tag{B.2}$$

and for the argument of the integral we can write

$$\begin{aligned} &f(x_i + \sigma s_i,\, \bar{x}_i + \sigma \bar{s}_i) - f(x_i + \sigma s_i,\, \bar{x}_i) = \\ &= \left(f(x_i + \sigma s_i,\, \bar{x}_i + \sigma \bar{s}_i) - f(x_i,\, \bar{x}_i) \right) - \left(f(x_i + \sigma s_i,\, \bar{x}_i) - f(x_i,\, \bar{x}_i) \right) \\ &= \nabla f(x')^T\, \sigma s - (\nabla f(x_i'', \bar{x}_i))_i\, \sigma\, s_i \\ &= (\nabla f(x'))_i\, \sigma\, s_i + \overline{(\nabla f(x'))_i}^T\, \sigma\, \bar{s}_i - (\nabla f(x_i'', \bar{x}_i))_i\, \sigma\, s_i \end{aligned} \tag{B.3}$$

with $x' \in (x, x+\sigma s)$ and $x_i'' \in (x_i, x_i+\sigma s_i)$.
We further have that

$$\overline{(\nabla f(x'))}_i = \overline{(\nabla f(x'))}_i - \overline{(\nabla f(x))}_i + \overline{(\nabla f(x))}_i \tag{B.4}$$

Now substituting (B.3) and (B.4) into (B.2) we obtain that

$$\begin{aligned} & g_\sigma(x_i,\, \bar{x}_i + \sigma\bar{s}_i) - g_\sigma(x_i,\, \bar{x}_i) = \\ & = \int_{-\infty}^{\infty} \left[(\nabla f(x'))_i - (\nabla f(x_i'', \bar{x}_i))_i\right]\; s_i^2\, \varphi(s_i)\, ds_i + \\ & + \int_{-\infty}^{\infty} \left(\overline{(\nabla f(x'))}_i - \overline{(\nabla f(x))}_i + \overline{(\nabla f(x))}_i\right)^T \bar{s}_i\, s_i\, \varphi(s_i)\, ds_i. \end{aligned} \tag{B.5}$$

By the Lipschitz property of the gradient, and recalling that

$$\int_{-\infty}^{\infty} \overline{(\nabla f(x))}_i^T\; \bar{s}_i\, s_i\, \varphi(s_i)\, ds_i = 0$$

we have:

$$\begin{aligned} & (g_\sigma(x_i,\, \bar{x}_i + \sigma\bar{s}_i) - g_\sigma(x_i,\, \bar{x}_i))^2 \leq \\ & \int_{-\infty}^{\infty} L^2\sigma^2\, \|s\|^2\, s_i^4\, \varphi(s_i)\, ds_i + \\ & \int_{-\infty}^{\infty} L^2\sigma^2\, \|s\|^2\, \|\bar{s}_i\|^2\, s_i^2\, \varphi(s_i)\, ds_i \end{aligned} \tag{B.6}$$

We can finally substitute (B.6) into (B.1) obtaining:

$$\begin{aligned} & \left((G_\sigma(x))_i - (\overline{G}_\sigma(x))_i\right)^2 \leq \\ & L^2\sigma^2 \int_{\mathbb{R}^{n-1}} \int_{-\infty}^{\infty} (s_i^2 + \|\bar{s}_i\|^2)\, s_i^4\, \varphi(s_i)\, ds_i\;\; \varphi(\bar{s}_i)\, d\bar{s}_i + \\ & + L^2\sigma^2 \int_{\mathbb{R}^{n-1}} \int_{-\infty}^{\infty} (s_i^2 + \|\bar{s}_i\|^2)\, \bar{s}_i{}^2\, \varphi(s_i)\, ds_i\;\; \varphi(\bar{s}_i)\, d\bar{s}_i. \end{aligned} \tag{B.7}$$

For the first term in (B.7) we obtain that

$$\begin{aligned} & \int_{\mathbb{R}^{n-1}} \int_{-\infty}^{\infty} (s_i^2 + \|\bar{s}_i\|^2)\, s_i^4\, \varphi(s_i)\, ds_i\;\; \varphi(\bar{s}_i)\, d\bar{s}_i \\ & = \int_{\mathbb{R}^{n-1}} \int_{-\infty}^{\infty} s_i^6\, \varphi(s_i)\, ds_i\;\; \varphi(\bar{s}_i)\, d\bar{s}_i \\ & \quad + \int_{\mathbb{R}^{n-1}} \int_{-\infty}^{\infty} \|\bar{s}_i\|^2\, s_i^4\, \varphi(s_i)\, ds_i\;\; \varphi(\bar{s}_i)\, d\bar{s}_i \\ & = 15 + 3(n-1). \end{aligned} \tag{B.8}$$

By similar computations the second term in (B.7) becomes

$$\begin{aligned} & \int_{\mathbb{R}^{n-1}} \int_{-\infty}^{\infty} (s_i^2 + \|\bar{s}_i\|^2)\, \|\bar{s}_i\|^2\, \varphi(s_i)\, ds_i\;\; \varphi(\bar{s}_i)\, d\bar{s}_i \\ & = \int_{\mathbb{R}^{n-1}} \int_{-\infty}^{\infty} s_i^2\, \|\bar{s}_i\|^2\, \varphi(s_i)\, ds_i\;\; \varphi(\bar{s}_i)\, d\bar{s}_i \\ & \quad + \int_{\mathbb{R}^{n-1}} \int_{-\infty}^{\infty} \|\bar{s}_i\|^4\, \varphi(s_i)\, ds_i\;\; \varphi(\bar{s}_i)\, d\bar{s}_i \\ & = (n-1) + 3(n-1) = 4(n-1). \end{aligned} \tag{B.9}$$

In (B.8) and (B.9) we used the property (page 208 in [34]) that for a zero mean Gaussian z with variance σ^2:

$$E[z^d] = \begin{cases} (d-1)!!\,\sigma^2, & \text{for } d \text{ even} \\ 0, & \text{for } d \text{ odd} \end{cases}$$

where $(d-1)\,!! = (d-1)(d-3)\cdots 3 \cdot 1$ and that for any $z \sim \mathcal{N}(0, I_{n-1})$

$$\int_{\mathbb{R}^{n-1}} \|z\|^2 \varphi(z)\, dz = \int_{\mathbb{R}^{n-1}} \sum_{i=1}^{n-1} z_i^2 \varphi(z)\, dz = n-1.$$

By substituting (B.8) and (B.9) in (B.7) we finally obtain that

$$\Big((G_\sigma(x))_i - (\overline{G}_\sigma(x))_i\Big)^2 \leq L^2\,\sigma^2\,(15 + 3(n-1) + 4(n-1)),$$

which, applied to all the entries, proves the theorem.

Appendix C

Brief notes on multi-objective optimization

In this Appendix we provide a very brief introduction on multi-objective optimization.

The main focus of this thesis is on single objective optimization problems like (2.1). There are some instances, though, where the objective function is constituted by multiple objectives, and the field that studies this kind of problems has the name of multi-objective (MO) optimization. For details on the algorithms and methods to approach this kind of problems the reader can refer to [90] and the references therein. Here we just give the definition of a multi-objective optimization problem and we outline two potential approaches to its solution.
A typical multi-objective optimization problem has the following structure:

$$\min_{x \in \mathbb{F}} (f_1(x), f_2(x), \ldots, f_k(x))^T \tag{C.1}$$

with $k \geq 2$ and $f_i : \mathbb{R}^n \to \mathbb{R}$ for $i = 1, \ldots, k$.
The general goal when solving problem (C.1) is the minimization of all the objective function $f_i(x)$ simultaneously. We indicate by $z_i, \quad i = 1, \ldots k$ a general solution to this problem.
If there were no conflicts between the objective functions, a trivial solution of the problem would be the one obtainable by solving k optimization problems separately (one for each objective function). In this scenario it would therefore not be necessary to apply any specific solution techniques. In most of the applications, though, this approach is infeasible, since the functions $f_1(x), f_2(x), \ldots, f_k(x)$ are, at least partially, in contrast with each other.
In this scenario, it becomes critical to understand what is an optimal solution of a multi-objective optimization problem. Let's start by reporting Pareto's 1896 definition [103]:

Definition 4. *Given two vectors $z^1 \in \mathbb{R}^k$ and $z^2 \in \mathbb{R}^k$, we say that z^1 Pareto*

dominates z^2 ($z^1 \leq_p z^2$) *if:*

$$\begin{aligned} &z_i^1 \leq z_i^2 \quad \forall i \in \{1, 2, \ldots, k\} \quad \textit{and} \\ &z_j^1 \leq z_j^2 \quad \textit{for at least one index} \quad j \in \{1, 2, \ldots, k\}. \end{aligned}$$

From this, the definition of *Pareto optimal solution* follows:

Definition 5. *A decision vector* $x^* \in \mathcal{F}$ *is a Pareto optimal solution if it does not exist another vector* $x \in \mathcal{F}$ *such that*

$$f(x) \leq_p f(x^*).$$

In other words, a solution is Pareto optimal if none of the objective functions can be improved without degrading some of the other objective values.
The set of the Pareto optimal solutions of a multi-objective optimization problem (MO) is called *Pareto frontier*.
We can now describe two different approaches for dealing with the presence of multiple objectives:

- algorithms with "a posteriori articulation of preferences", where the optimal solution is provided in terms of Pareto front;
- scalarization techniques, that consist in reducing the multi-objective optimization problem to a single objective one by considering the weighted sum of the objective functions and by performing several minimizations, with different combinations of the weights, thus obtaining an approximate Pareto front.

The vast world of the multi-objective optimization is outside the scope of this thesis. It is sufficient to say that it is always possible to reduce any MO optimization problem to a series of single objective problems, each of which can be solved using the standard algorithms and techniques for this class of problems. Clearly, the higher the number of combinations of weights considered, the higher the computation cost for the resolution of the problem.

Bibliography

[1] Streaming care: Fast track services in hospital emergency departments. Metropolitan Health and Aged Care Services Division, Victorian Government Department of Human Services, Discussion paper, 2008.

[2] C. C. Aggarwal. *Data mining: the textbook*. Springer, 2015.

[3] V. Ahalt, N. Argon, J. Strickler, and A. Mehrotra. Comparison of emergency department crowding scores: a discrete-event simulation approach. *Health Care Management Science*, 21:144–155, 2018.

[4] M. A. Ahmed and T. M. Alkhamis. Simulation optimization for an emergency department healthcare unit in Kuwait. *European Journal of Operational Research*, 198(3):936 – 942, 2009.

[5] K. Ahsan, M. Alam, D. Morel, and M. Karim. Emergency department resource optimisation for improved performance: a review. *Journal of Industrial Engineering International*, 15 (Supp 1):S253–S266, 2019.

[6] E. J. Anderson and M. C. Ferris. A direct search algorithm for optimization with noisy function evaluations. *SIAM Journal on optimization*, 11(3):837–857, 2001.

[7] R. Aringhieri, G. Bonetta, and D. Duma. Reducing overcrowding at the emergency department through a different physician and nurse shift organisation: a case study. In P. Daniele and L. Scrimali, editors, *New Trends in Emergency Complex Real Life Problems*, volume 1 of *AIRO Springer Series*, pages 43–53. Springer Nature Switzerland, 2018.

[8] R. Aringhieri, M. Bruni, S. Khodaparasti, and J. van Essen. Emergency medical services and beyond: Addressing new challenges through a wide literature review. *Computers & Operations Research*, 78:349 – 368, 2017.

[9] K. E. Atkinson. *An Introduction to Numerical Analysis*. John Wiley & Sons, 2008.

[10] C. Audet and W. Hare. *Derivative-Free and Blackbox Optimization*. Springer International Publishing, 01 2017.

[11] C. Audet, G. Savard, and W. Zghal. Multiobjective optimization through a series of single-objective formulations. *SIAM J. on Optimization*, 19(1):188–210, Feb. 2008.

[12] K. Balasubramanian and S. Ghadimi. Zeroth-order nonconvex stochastic optimization: Handling constraints, high dimensionality, and saddle points. *Foundations of Computational Mathematics*, pages 1–42, 2021.

[13] J. Barzilai and J. M. Borwein. Two-point step size gradient methods. *IMA journal of numerical analysis*, 8(1):141–148, 1988.

[14] L. Bedoya-Valencia and E. Kirac. Evaluating alternative resource allocation in an emergency department using discrete event simulation. *Simulation*, 92:1041–1051, 2016.

[15] A. S. Berahas, R. H. Byrd, and J. Nocedal. Derivative-free optimization of noisy functions via quasi-newton methods. *SIAM Journal on Optimization*, 29(2):965–993, 2019.

[16] A. S. Berahas, L. Cao, K. Choromanski, and K. Scheinberg. Linear interpolation gives better gradients than gaussian smoothing in derivative-free optimization. *arXiv preprint arXiv:1905.13043*, 2019.

[17] A. S. Berahas, L. Cao, K. Choromanski, and K. Scheinberg. A theoretical and empirical comparison of gradient approximations in derivative-free optimization. *Foundations of Computational Mathematics*, pages 1–54, 2021.

[18] S. Bernstein, V. Verghese, W. Leung, A. T Lunney, and I. Perez. Development and validation of a new index to measure emergency department crowding. *Academic emergency medicine: official journal of the Society for Academic Emergency Medicine*, 10:938–42, 10 2003.

[19] D. Berrar. Cross-validation., 2019.

[20] D. P. Bertsekas. Nonlinear programming. *Journal of the Operational Research Society*, 48(3):334–334, 1997.

[21] M. Boresta, T. Colombo, A. De Santis, and S. Lucidi. A mixed finite differences scheme for gradient approximation. *Journal of Optimization Theory and Applications*, pages 1–24, 2022.

[22] L. Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer, 2010.

[23] J. P. Boyd. *Chebyshev and Fourier Spectral Methods*. Springer-Verlag Berlin Heidelberg, 2001.

[24] S. Brailsford, P. Harper, B. Patel, and M. Pitt. An analysis of the academic literature on simulation and modelling in health care. *Journal of Simulation*, 3:130–140, 2009.

[25] M. D. Buhmann. *Radial basis functions: theory and implementations*, volume 12. Cambridge university press, 2003.

[26] T.-L. Chen and C.-C. Wang. Multi-objective simulation optimization for medical capacity allocation in emergency department. *Journal of Simulation*, 10(1):50–68, 2016.

[27] J. Cho and K. D. Dorfman. Brownian dynamics simulations of electrophoretic dna separations in a sparse ordered post array. *Journal of Chromatography A*, 1217(34):5522–5528, 2010.

[28] M. Claesen and B. De Moor. Hyperparameter search in machine learning. *arXiv preprint arXiv:1502.02127*, 2015.

[29] G. Cocchi, G. Liuzzi, A. Papini, and M. Sciandrone. An implicit filtering algorithm for derivative-free multiobjective optimization with box constraints. *Computational Optimization and Applications*, 69(2):267–296, March 2018.

[30] A. Conn, K. Scheinber, and L. Vicente. *Introduction to Derivative-Free Optimization*. Society for Industrial and Applied Mathematics, Philadelphia, 2009.

[31] A. R. Conn, K. Scheinberg, and L. N. Vicente. *Introduction to derivative-free optimization*. SIAM, 2009.

[32] J. Considine, M. Kropman, E. Kelly, and C. Winter. Effect of emergency department fast track on emergency department length of stay: a case–control study. *Emergency Medical Journal*, 25:815–819, 2008.

[33] J. Copeland and A. Gray. A daytime fast track improves throughput in a single physician coverage emergency department. *CJEM*, 17 6:648–55, 2015.

[34] H. Cramér. *Mathematical Methods of Statistics*, volume 43. Princeton University Press, 1999.

[35] A. L. Custódio, J. Madeira, A. Vaz, and L. Vicente. Direct multisearch for multiobjective optimization. *SIAM J. Optim.*, 21:1109–1140, 2011.

[36] G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989.

[37] Y.-H. Dai and L.-Z. Liao. R-linear convergence of the Barzilai and Borwein gradient method. *IMA Journal of Numerical Analysis*, 22(1):1–10, 2002.

[38] D. Daldoul, I. Nouaouri, H. Bouchriha, and H. Allaoui. Optimization on human and material resources in emergency department. In *2015 International Conference on Industrial Engineering and Systems Management (IESM)*, pages 633–638, 2015.

[39] D. Daldoul, I. Nouaouri, H. Bouchriha, and H. Allaoui. A stochastic model to minimize patient waiting time in an emergency department. *Operations Research for Health Care*, 18:16 – 25, 2018. EURO 2016–New Advances in Health Care Applications.

[40] A. De Santis, T. Giovannelli, S. Lucidi, M. Messedaglia, and M. Roma. A simulation-based optimization approach for the calibration of a discrete event simulation model of an Emergency Department. *Annals of Operations Research*, 2021.

[41] A. De Santis, T. Giovannelli, S. Lucidi, M. Messedaglia, and M. Roma. Determining the optimal piecewise constant approximation for the nonhomogeneous Poisson process rate of Emergency Department patient arrivals. *Flexible Services and Manufacturing Journal*, 2021.

[42] A. De Santis, T. Giovannelli, S. Lucidi, M. Messedaglia, and M. Roma. A simulation-based optimization approach for the calibration of a discrete event simulation model of an emergency department. *arXiv preprint arXiv:2102.00945*, 2021.

[43] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *Trans. Evol. Comp*, 6(2):182–197, Apr. 2002.

[44] M. Ehrgott. *Multicriteria Optimization.* Springer-Verlag, Berlin, Heidelberg, 2005.

[45] G. Fava, T. Giovannelli, M. Messedaglia, and M. Roma. Effect of different patient peak arrivals on an emergency department via discrete event simulation: a case study. *Simulation: Transactions of the Society for Modeling and Simulation International*, 2021.

[46] Y.-Y. Feng, I.-C. Wu, and T.-L. Chen. Stochastic resource allocation in emergency departments with a multi-objective simulation optimization algorithm. *Health care management science*, 20, 08 2015.

[47] A. D. Flaxman, A. T. Kalai, and H. B. McMahan. Online convex optimization in the bandit setting: gradient descent without a gradient. *arXiv preprint cs/0408007*, 2004.

[48] J. Friedman, T. Hastie, R. Tibshirani, et al. *The elements of statistical learning*, volume 1. Springer series in statistics New York, 2001.

[49] M. C. Fu, editor. *Handbook of Simulation Optimization*. Springer, New York, 2015.

[50] I. M. Gel'fand and G. E. Shilov. *Generalized Functions, Volume 2: Spaces of Fundamental and Generalized Functions*, volume 261. American Mathematical Soc., 2016.

[51] N. Gilboy, P. Tanabe, D. Travers, and A. Rosenau. Emergency Severity Index. A Triage Tool for Emergency Department Care – Implementation Handbook, 2020 edition. Emergency Nurses Association (ENA), 2020.

[52] I. Goodfellow, Y. Bengio, and A. Courville. *Deep learning*. MIT press, 2016.

[53] A. Gosavi. *Simulation-Based Optimization: Parametric Optimization Techniques and Reinforcement Learning*. Springer Publishing Company, Incorporated, 2nd edition, 2014.

[54] O. Granichin. Stochastic approximation with input perturbation under dependent observation noises. *Bulletin of the Leningrad University. Series 1: Mathematics, Mechanics, Astronomy*, (4):27–31, 1989.

[55] P. Graven. Smoothing noisy data with spline function: estimating the correct degree of smoothing by the method of generalized cross-validaton. *Number. Math.*, 31:377–403, 1989.

[56] L. Grippo and M. Sciandrone. Nonmonotone globalization techniques for the barzilai-borwein gradient method. *Computational Optimization and Applications*, 23(2):143–169, 2002.

[57] L. Grippo and M. Sciandrone. *Metodi di ottimizzazione non vincolata*. Springer Science & Business Media, 2011.

[58] M. Gul and A. F. Guneri. A comprehensive review of emergency department simulation applications for normal and disaster conditions. *Comput. Ind. Eng.*, 83(C):327–344, May 2015.

[59] H. Guo, S. Gao, K. Tsui, and T. Niu. Simulation optimization for medical staff configuration at emergency department in Hong Kong. *IEEE Transactions on Automation Science and Engineering*, 14(4):1655–1665, Oct 2017.

[60] H. Guo, D. Goldsman, K.-L. Tsui, Y. Zhou, and S.-Y. Wong. Using simulation and optimisation to characterise durations of emergency department service times with incomplete data. *International Journal of Production Research*, 54(21):6494–6511, 2016.

[61] M. T. Hagan, H. B. Demuth, and M. Beale. *Neural network design*. PWS Publishing Co., 1997.

[62] H. Hajjarsaraei, B. Shirazi, and J. Rezaeian. Scenario-based analysis of fast track strategy optimization on emergency department using integrated safety simulation. *Safety Science*, 107:9 – 21, 2018.

[63] D. M. Hawkins. The problem of overfitting. *Journal of chemical information and computer sciences*, 44(1):1–12, 2004.

[64] N. Hoot and D. Aronsky. Systematic review of emergency department crowding: causes, effects, and solutions. *Annals of Emergency Medicine*, 52(2):126–136, 2008.

[65] N. R. Hoot, C. H. Zhou, I. Jones, and D. Aronsky. Measuring and forecasting emergency department crowding in real time. *Annals of emergency medicine*, 49 6:747–55, 2007.

[66] K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

[67] T. J Reeder, D. Burleson, and H. G Garrison. The overcrowded emergency department: A comparison of staff perceptions. *Academic emergency medicine: official journal of the Society for Academic Emergency Medicine*, 10:1059–64, 11 2003.

[68] V. Joshi, C. Lim, and S. G. Teng. Simulation study: Improvement for non-urgent patient processes in the emergency department. *Engineering Management Journal*, 28:3:145–157, 2016.

[69] B. Karlik and A. V. Olgac. Performance analysis of various activation functions in generalized mlp architectures of neural networks. *International Journal of Artificial Intelligence and Expert Systems*, 1(4):111–122, 2011.

[70] A. Kaushal, Y. Zhao, Q. Peng, T. Strome, E. Weldon, M. Zhang, and A. Chochinov. Evaluation of fast track strategies using agent-based simulation modeling to reduce waiting time in a hospital emergency department. *Socio-Economic Planning Sciences*, 50:18–31, 2015.

[71] S.-H. Kim and W. Whitt. Are call center and hospital arrivals well modeled by nonhomogeneous Poisson process ? *Manufactory & Service Operations Management*, 16:464–480, 2014.

[72] S.-H. Kim and W. Whitt. The power of alternative Kolmogorov–Smirnov tests based on transformations of the data. *ACM Transactions on Modeling and Computer Simulation*, 25(4):1–22, 2015.

[73] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[74] A. Kleywegt, A. Shapiro, and T. Homem–de–Mello. The sample average approximation method for stochastic discrete optimization. *SIAM Journal on Optimization*, 12:479–502, 2001.

[75] D. Kraft et al. A software package for sequential quadratic programming. 1988.

[76] A. Kramer, C. Dosi, M. Iori, and M. Vignoli. Successful implementation of discrete event simulation: the case of an Italian emergency department, 2020.

[77] Y.-H. Kuo, J. M. Leung, C. A. Graham, K. K. Tsoi, and H. M. Meng. Using simulation to assess the impacts of the adoption of a fast-track system for hospital emergency services. *Journal of Advanced Mechanical Design, Systems, and Manufacturing*, 12(3):1–11, 2018.

[78] Y.-H. Kuo, O. Rado, B. Lupia, J. M. Y. Leung, and C. A. Graham. Improving the efficiency of a hospital emergency department: a simulation study with indirectly imputed service-time distributions. *Flexible Services and Manufacturing Journal*, 28(1):120–147, Jun 2016.

[79] P. Landa, M. Sonnessa, E. Tànfani, and A. Testi. Multiobjective bed management considering emergency and elective patient flows. *International Transactions in Operational Research*, 25(1):91–110, 2018.

[80] Z. Liu, E. Cabrera, M. Taboada, F. Epelde, D. Rexachs, and E. Luque. Quantitative evaluation of decision effects in the management of emergency department problems. *Procedia Computer Science*, 51:433–442, 2015.

[81] G. Liuzzi, S. Lucidi, F. Parasiliti, and M. Villani. Multiobjective optimization techniques for the design of induction motors. *IEEE Transactions on Magnetics*, 39(3):1261–1264, 2003.

[82] G. Liuzzi, S. Lucidi, and F. Rinaldi. A derivative-free approach to constrained multiobjective nonsmooth optimization. *SIAM Journal on Optimization*, 26(4):2744–2774, 2016.

[83] G. Liuzzi, S. Lucidi, and F. Rinaldi. An algorithmic framework based on primitive directions and nonmonotone line searches for black-box optimization problems with integer variables. *Mathematical Programming Computation*, 02 2020.

[84] S. Lucidi and M. Sciandrone. On the global convergence of derivative-free methods for unconstrained optimization. *SIAM Journal on Optimization*, 13(1):97–116, 2002.

[85] A. Maggiar, A. Wachter, I. S. Dolinskaya, and J. Staum. A derivative-free trust-region algorithm for the optimization of functions smoothed via gaussian

convolution using adaptive multiple importance sampling. *SIAM Journal on Optimization*, 28(2):1478–1507, 2018.

[86] R. Marler and J. Arora. Survey of multi-objective optimization methods for engineering. *Structural and Multidisciplinary Optimization*, 26:369–395, 2004.

[87] W. S. McCulloch and W. Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 1943.

[88] R. C. Merton. On the pricing of corporate debt: The risk structure of interest rates. *The Journal of finance*, 29(2):449–470, 1974.

[89] K. Miettinen. *Nonlinear Multiobjective Optimization*. Kluwer Academic Publishers, 1999.

[90] K. Miettinen. *Nonlinear multiobjective optimization*, volume 12. Springer Science & Business Media, 2012.

[91] J. J. Moré and S. M. Wild. Estimating computational noise. *SIAM Journal on Scientific Computing*, 33(3):1292–1314, 2011.

[92] J. J. Moré and S. M. Wild. Estimating derivatives of noisy simulations. *ACM Transactions on Mathematical Software (TOMS)*, 38(3):1–21, 2012.

[93] C. Morley, M. Unwin, J. Peterson, Gregory M andStankovich, and L. Kinsman. Emergency department crowding: A systematic review of causes, consequences and solutions. *PLoS One*, 13, 08 2018.

[94] A. Nahhas, A. Alwadi, and T. Reggelin. Simulation and the emergency department overcrowding problem. *Procedia Engineering*, 178:368–376, 12 2017.

[95] V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Icml*, 2010.

[96] A. Nedić and D. P. Bertsekas. The effect of deterministic noise in subgradient methods. *Mathematical programming*, 125(1):75–99, 2010.

[97] J. A. Nelder and R. Mead. A simplex method for function minimization. *The computer journal*, 7(4):308–313, 1965.

[98] Y. Nesterov and V. Spokoiny. Random gradient-free minimization of convex functions. *Foundations of Computational Mathematics*, 17(2):527–566, 2017.

[99] J. Nocedal and S. Wright. *Numerical optimization*. Springer Science & Business Media, 2006.

[100] J. Nocedal and S. J. Wright. Sequential quadratic programming. *Numerical optimization*, pages 529–562, 2006.

[101] C. Nwankpa, W. Ijomah, A. Gachagan, and S. Marshall. Activation functions: Comparison of trends in practice and research for deep learning. *arXiv preprint arXiv:1811.03378*, 2018.

[102] U. M. G. Palomares. A unified convergence theory for non monotone direct search methods (dsms) with extensions to dfo with mixed and categorical variables. 2019.

[103] V. Pareto. *Cours d'économie politique*, volume 1. Librairie Droz, 1964.

[104] S. A. Paul, M. C. Reddy, and C. J. De Flitch. A systematic review of simulation studies investigating emergency department overcrowding. *Simulation*, 86(8-9):559–571, 2010.

[105] V. Picheny, D. Ginsbourger, O. Roustant, R. T. Haftka, and N.-H. Kim. Adaptive designs of experiments for accurate approximation of a target region. 2010.

[106] J. M. Pines, S. Iyer, M. Disbot, J. E. Hollander, F. S. Shofer, and E. M. Datner. The effect of emergency department crowding on patient satisfaction for admitted patients. *Academic Emergency Medicine*, 15(9):825–831, 2008.

[107] A. Pinkus. Approximation theory of the mlp model in neural networks. *Acta numerica*, 8:143–195, 1999.

[108] E. Plambeck, B. Fu, S. Robinson, and R. Suri. Sample–path optimization of convex stochastic performance functions. *Mathematical Programming*, 75:137–176, 1996.

[109] M. J. Powell. Unconstrained minimization algorithms without computation of derivatives. 1974.

[110] P. Ramachandran, B. Zoph, and Q. V. Le. Searching for activation functions. *arXiv preprint arXiv:1710.05941*, 2017.

[111] M. Raydan. The barzilai and borwein gradient method for the large scale unconstrained minimization problem. *SIAM Journal on Optimization*, 7(1):26–33, 1997.

[112] S. Robinson. Analysis of sample–path optimization. *Mathematics of Operations Research*, 21:513–528, 1996.

[113] F. Rosenblatt. Principles of neurodynamics. perceptrons and the theory of brain mechanisms. Technical report, Cornell Aeronautical Lab Inc Buffalo NY, 1961.

[114] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

[115] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

[116] S. Saghafian, G. Austin, and S. Traub. Operations research/management contributions to emergency department patient flow optimization: Review and research prospects. *IIE Transactions on Healthcare Systems Engineering*, 5:101–123, 2015.

[117] T. Salimans, J. Ho, X. Chen, S. Sidor, and I. Sutskever. Evolution strategies as a scalable alternative to reinforcement learning. *arXiv preprint arXiv:1703.03864*, 2017.

[118] A. Salmon, S. Rachuba, S. Briscoe, and M. Pitt. A structured literature review of simulation modelling applied to emergency departments: Current patterns and emerging trends. *Operations Research for Health Care*, 19:1–13, 2018.

[119] W. S. Sarle et al. Stopped training and other remedies for overfitting. *Computing science and statistics*, pages 352–360, 1996.

[120] K. Schittkowski. *More test examples for nonlinear programming codes*, volume 282. Springer Science & Business Media, 2012.

[121] M. Schull, A. Kiss, and J.-P. Szalai. The effect of low-complexity patients on emergency department waiting times. *Annals of emergency medicine*, 49 3:257–64, 264.e1, 2007.

[122] H.-J. M. Shi, M. Q. Xuan, F. Oztoprak, and J. Nocedal. On the numerical performance of derivative-free optimization methods based on finite-difference approximations. *arXiv preprint arXiv:2102.09762*, 2021.

[123] B. Shirazi. Fast track system optimization of emergency departments: Insights from a computer simulation study. *International Journal of Modeling, Simulation, and Scientific Computing*, 07, 02 2016.

[124] J. C. Spall. A one-measurement form of simultaneous perturbation stochastic approximation. *Automatica*, 33(1):109–112, 1997.

[125] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

[126] I. Sutskever, J. Martens, G. Dahl, and G. Hinton. On the importance of initialization and momentum in deep learning. In *International conference on machine learning*, pages 1139–1147. PMLR, 2013.

[127] H. Tran and G. Zhang. Adadgs: An adaptive black-box optimization method with a nonlocal directional gaussian smoothing gradient. *arXiv preprint arXiv:2011.02009*, 2020.

[128] I. Ucar, B. Smeets, and A. Azcorra. simmer: Discrete-event simulation for R. *Journal of Statistical Software*, 90(2):1–30, 2019.

[129] A. Uriarte, E. Zúñiga, M. U. Moris, and A. Ng. How can decision makers be supported in the improvement of an emergency department? : A simulation, optimization and data mining approach. *Operations research for health care*, 15:102–122, 2017.

[130] A. G. Uriarte, E. R. Zúñiga, M. U. Moris, and A. H. C. Ng. System design and improvement of an emergency department using simulation-based multi-objective optimization. *Journal of Physics: Conference Series*, 616:012015, may 2015.

[131] L. Vanbrabant, K. Braekers, K. Ramaekers, and I. Van Nieuwenhuyse. Simulation of emergency department operations: A comprehensive review of KPIs and operational improvements. *Computer & Industrial Engineering*, 131:356–383, 2019.

[132] V. Vapnik. *The nature of statistical learning theory*. Springer science & business media, 1999.

[133] H. Wang, R. D. Robinson, J. S. Garrett, K. Bunch, C. A. Huggins, K. Watson, J. Daniels, B. Banks, J. P. D'Etienne, and N. R. Zenarosa. Use of the SONET score to evaluate high volume emergency department overcrowding: A prospective derivation and validation study. *Emergency Medicine International*, 11:1–11, 2015.

[134] S. Weiss, R. Derlet, J. Arndahl, A. Ernst, J. Richards, M. Fernández-Frankelton, R. Schwab, T. Stair, P. Vicellio, D. Levy, M. Brautigan, A. Johnson, and T. Nick. Estimating the degree of emergency department overcrowding in academic medical centers: Results of the national ed overcrowding study (NEDOCS). *Academic Emergency Medicine*, 11(1):38–50, 1 2004.

[135] S. Weiss, A. A. Ernst, and T. G. Nick. Comparison of the national emergency department overcrowding scale and the emergency department work index for quantifying emergency department crowding. *Academic emergency medicine: official journal of the Society for Academic Emergency Medicine*, 13 5:513–8, 2006.

[136] D. Wierstra, T. Schaul, T. Glasmachers, Y. Sun, J. Peters, and J. Schmidhuber. Natural evolution strategies. *The Journal of Machine Learning Research*, 15(1):949–980, 2014.

[137] S. M. Wild, R. G. Regis, and C. A. Shoemaker. Orbit: Optimization by radial basis function interpolation in trust-regions. *SIAM Journal on Scientific Computing*, 30(6):3197–3219, 2008.

[138] J. Wiler, R. Griffey, and T. Olsen. Review of modeling approaches for emergency department patient flow and crowding research. *Academic Emergency Medicine*, 18:1371–1379, 2011.

[139] Z. S.-Y. Wong, A. C.-H. Lit, S.-Y. Leung, K.-L. Tsui, and K.-S. Chin. A discrete-event simulation study for emergency room capacity management in a hong kong hospital. *2016 Winter Simulation Conference (WSC)*, pages 1970–1981, 2016.

[140] M. H. Wright. Direct search methods: Once scorned, now respectable. *Pitman Research Notes in Mathematics Series*, pages 191–208, 1996.

[141] M. Yarmohammadian, F. Rezaei, A. Haghshenas1, and N. Tavakoli. Overcrowding in emergency departments: A review of strategies to decrease future challenges. *Journal of Research in Medical Sciences*, 22:23, 2017.

[142] R. Yazdanparast, M. Hamid, M. A. Azadeh, and A. Keramati. An intelligent algorithm for optimization of resource allocation problem by considering human error in an emergency department. *Journal of Industrial and Systems Engineering*, 11(1):287–309, 2018.

[143] M. Yousefi, M. Yousefi, and F. S. Fogliatto. Simulation-based optimization methods applied in hospital emergency departments: A systematic review. *Simulation: Transactions of the Society for Modeling and Simulation International*, 96(10):791–806, 2020.

[144] F. Zeinali, M. Mahootchi, and M. Sepehri. Resource planning in the emergency departments: a simulation-base metamodeling approach. *Simulation Modelling Practice and Theory*, 53:123–138, 2015.

[145] J. Zhang, H. Tran, D. Lu, and G. Zhang. A novel evolution strategy with directional gaussian smoothing for blackbox optimization. *arXiv preprint arXiv:2002.03001*, 2020.

[146] X. Zuo, B. Li, X. Huang, M. Zhou, C. Cheng, X. Zhao, and Z. Liu. Optimizing hospital emergency department layout via multiobjective tabu search. *IEEE Transactions on Automation Science and Engineering*, 16:1137–1147, 2019.