Surrogate-Based Methods (SBMs) are algorithms employed for Black-Box and Preference-Based Optimization (BBO and PBO). In BBO, the objective function is unknown and expensive to evaluate. In PBO, information comes in the form of preferences: a decision-maker compares calibrations stating, e.g., "This tuning is better than that one". BBO and PBO problems may also include constraints that are unknown and expensive to evaluate. SBMs iteratively propose new calibrations to try by trading off exploitation of the surrogates (i.e. approximations of the unknown functions) and exploration of the decision space. This book extends four recent SBMs with the following results: global convergence guarantees, increased robustness and efficiency, and successful application to a control systems case study.

Davide Previtali earned a PhD in Engineering and Applied Sciences (35th Cycle) at the University of Bergamo, where he currently is Assistant Professor, working in the Control systems and Automation Laboratory. His research activity covers three main macro-areas: control systems, black-box and preference-based optimization, and lithium-ion batteries.

ISBN: 978-88-97413-93-6
DOI: 10.13122/978-88-97413-93-6

Davide Previtali SURROGATE-BASED OPTIMIZATION 70

Collana della Scuola di Alta Formazione Dottorale
- 70 -

Davide Previtali

SURROGATE-BASED OPTIMIZATION in control systems

UNIVERSITÀ DEGLI STUDI DI BERGAMO

2024

Collana della Scuola di Alta Formazione Dottorale

- 70 -

Collana della Scuola di Alta Formazione Dottorale

Diretta da Paolo Cesaretti

Ogni volume è sottoposto a *blind peer review*.

ISSN: 2611-9927

Sito web: https://aisberg.unibg.it/handle/10446/130100

Davide Previtali

SURROGATE-BASED METHODS FOR BLACK-BOX and preference-based optimization in control systems

Università degli Studi di Bergamo

2024

Surrogate-based methods for black-box and preference-based optimization in control systems / Davide Previtali. – Bergamo : Università degli Studi di Bergamo, 2024.
(Collana della Scuola di Alta Formazione Dottorale; 70)

ISBN: 978-88-97413-93-6
DOI: 10.13122/978-88-97413-93-6

Progetto grafico: Servizi Editoriali – Università degli Studi di Bergamo
© 2018 Università degli Studi di Bergamo
via Salvecchio, 19
24129 Bergamo
Cod. Fiscale 80004350163
P. IVA 01612800167

https://aisberg.unibg.it/handle/10446/282470

Acknowledgments

My sincere thanks go to the members of my research group, the Control systems and Automation Laboratory at the University of Bergamo, with whom I have shared these past three years of my doctorate. Together we have faced many failures and successes, supporting each other nonetheless. In no particular order, I would like to thank:

Nicholas, who is always reliable and never fails to get things done;

Leandro, with whom I have shared many endeavors and emerged victorious;

Marco, whose positivity and brightness are contagious;

Davide C., with his impeccable sense of style;

Luca, who taught me that passions are what make a man;

The newest additions to my research group, Pietro and Beatrice. I am sure you will do great.

Special thanks go to my supervisors, Prof. Mirko Mazzoleni, Prof. Antonio Ferramosca and Prof. Fabio Previdi, who have taught me several valuable lessons, scientifically and whatnot. Your bits of advice and teachings made me a better man.
I warmly thank my closest friends, Sergio, Christian and Matteo, who have always been supportive and understanding in these challenging years. My gratitude also goes to all new acquaintances made in these past years, who are happily too many to list here. I tightly hold on to all the laughs and the hardships that we have shared.
Lastly, my sincerest thanks go to my family for their unwavering support. You have always believed in me, even in my worst moments.

Table of Contents

Introduction

In many applications there is the need to calibrate a vector of *decision variables*. For instance, in the context of control systems, we might be interested in tuning the parameters of a controller in order to achieve some desired performances [36]. Similarly, many machine learning methods require a proper calibration of their hyper-parameters so that certain requirements are met [15]. An important remark is that, in some cases, it might be difficult or even impossible to objectively quantify the “goodness” of a calibration of the decision variables. For example, the performances of a controller tuning might be assessed by a human operator (decision-maker) that expresses his/her judgement through visual inspection (or other sensory evaluations) of the behavior achieved by the system under control. Whether the “goodness” of a calibration is evaluated subjectively or objectively, we are often interested in finding the tuning that attains the best results, i.e. the one that maximizes a measure of performance, minimizes some cost or best satisfies the human decision-maker’s criterion. At the same time, the values that the decision variables can assume might be restricted in some way. For example, only those controller calibrations that stabilize the system under control [102] are actually of interest. The goal that we have in mind is concise, yet staggeringly difficult: *find the global solution(s) of an optimization problem* [143]. The “goodness” of a calibration is described by an *objective function*[1], which is either minimized or maximized. The *constraints* of the optimization problem distinguish the tunings that are allowed (*feasible*) from those that are not (*infeasible*), effectively defining the *decision space*. In most cases, it is also possible to quantify to which degree each constraint is satisfied/violated. Finally, a *global solution* of an optimization problem is a feasible calibration of the decision variables that achieves the lowest (or highest) value of the objective function.
Numerous types of optimization problems exist. We could distinguish between linear and nonlinear [100], convex and non-convex [18], local and global [143] optimization problems. Alternatively, motivated by the previous examples, we could classify an optimization problem based on the information that we have available on the objective function and the constraints functions. We distinguish between:

- *A-priori known* objective/constraints functions, in a sense that their analytical expressions are available. For example, we might be interested in estimating the parameters (decision vector) of a model that describes some physical phenomenon. To do so, we could acquire some data on the phenomenon and find the parametrization that minimizes the deviation, in a Least Squares [57] sense, between the model and the observations (objective function). Some parameters might

[1]Objective as in goal, not to be confused as the opposite of subjective. In fact, as we will see in this book, an objective function can represent the subjective criterion of a human decision-maker.

represent positive physical quantities (such as masses, resistances, and so on), hence we require a (a-priori known) positivity constraint for each.

- *Unknown but measurable* objective/constraints functions. We could view these as *black-boxes* that take a calibration of the decision vector as the input and return the degree of "goodness"/violation as the output. For instance, we might be interested in finding the controller calibration that minimizes the settling time of the closed-loop step response of a control system but, at the same time, exhibits a small overshoot [36]. Typically, there does not exist a tuning that simultaneously minimizes both criteria, a trade-off must be made [84]. We could proceed as follows: minimize the settling time (objective function) and ensure that the overshoot does not exceed a maximum tolerated level (constraint). Clearly, if no accurate mathematical model of the control system is available, the settling time and the overshoot resulting from a calibration of the controller's parameters can only be measured by performing an experiment.

- *Unknown and not quantifiable objectively* objective/constraints functions. In this case, the optimization is carried out by actively querying a human *decision-maker*, who discerns "good" calibrations from "bad" ones (objective function) and, at the same time, assesses whether a tuning is acceptable (feasible) or not (infeasible). In this case, information on the objective function is expressed in the form of *preferences*, e.g. "this calibration is better than that one". From an *utility theory* [104] standpoint, the preferences of the decision-maker can be described by a (unknown) mathematical function that associates to each tuning its level of satisfaction (*utility*). Furthermore, the most preferred tuning is the one which has the highest utility. Constraints that are unknown and cannot be quantified objectively, which we refer to as *decision-maker-based constraints*, are assessed in a different manner: we ask the decision-maker a "yes/no question" (e.g. "is this calibration acceptable?") [156]. From an optimization perspective, acceptability is equivalent to feasibility. We wrap up this last classification of the objective/constraints functions with a clarifying example. Consider a highly configurable packaging plant and suppose that the quality of the produced packages depends on several calibration parameters (decision vector). A quality control inspector assesses the performances of the plant by taking samples (packages) resulting from different tunings of the industrial process. The inspector would like to select the calibration that best suits his/her quality standards (e.g. the packages are wrapped uniformly, the end product is visually appealing, and so on). To do so, he/she keeps comparing samples resulting from different parametrizations (such as "the packages obtained by calibration $\boldsymbol{x}_1$ are better than those produced by $\boldsymbol{x}_2$"), until a satisfactory tuning is found. At the same time, it is reasonable to assume that some calibrations of the packaging plant produce undesirable results (e.g. if some of the packages are damaged in the process). In such cases, the inspector simply states that the tuning is not acceptable (infeasible)[2].

In practice, an optimization problem could include a combination of different types of objective/-constraints functions. For example, we could have problems with a black-box objective function but a-priori known constraints functions.

The task of finding the global solutions of an optimization problem has been addressed in the *global optimization* literature [59, 97, 139, 143]. Compared to local optimization [100], finding global solutions is much harder due to the fact that an exhaustive search of the feasible region might be required. Nonetheless, many general purpose global optimization algorithms have been proposed in the past fifty years. These include: DIvide a hyper-RECTangle (`DIRECT` [67]), multi-start methods [90], simulated annealing [75], Particle SWARM (`PSWARM` [72]) optimization and evolutionary algorithms (such as genetic algorithms) [8]. In practice, most of these methods require an excruciatingly high number of evaluations of the objective/constraints functions to converge to a sufficiently accurate global solution of the optimization problem. Clearly, whenever either or both the objective function and the constraints functions are unknown and *expensive* to measure (e.g. they might require running a computer simulation or performing a real-world experiment, see previous examples), the number of evaluations must be as low as possible. Thus, a more suitable and realistic goal would be: *find a calibration of the decision vector with a lower degree of optimality accuracy but using the least amount of evaluations*. Driven by this goal, two branches of global optimization have emerged: *black-box optimization* and *preference-based optimization*. The former addresses those global optimization problems wherein the objective function and (possibly) the constraints functions are unknown but measurable, while the latter handles calibration processes which involve human decision-makers. Black-box optimization has gained a lot of popularity in the past two decades due to the widespread of *Bayesian optimization* [22, 45, 93], which has been successfully used to tune the hyper-parameters of machine learning algorithms [134] but also for controller calibration purposes [14, 42, 73, 89, 98]. At the moment, preference-based optimization is more niche, very few methods [11, 12, 21, 52, 156] and applications [120, 155] are present in the literature.

[2]In this example, the objective function (quality of the packages) and the constraints (acceptability) are closely related: an acceptable tuning is “better than” any unacceptable one. In practice, this need not be the case. For example, we could have two decision-makers: the quality inspector, who defines the “goodness” of the packages, and an industrial operator, who checks if the plant is working properly. It could be that the tunings which produce high quality packages are too “demanding” for the industrial process (unacceptable).

The most widely used algorithms for black-box and preference-based optimization are *surrogate-based methods* (or *response surface techniques*) [65, 149]. These methods rely on approximations of the objective function and the constraints functions, which are referred to as *surrogates*, that are *cheap* to evaluate (as opposed to their counterparts). The rationale behind response surface techniques is quite straightforward:

1. Approximate the objective function and the constraints functions using only the information at hand (i.e. the measures and/or the feedback from the decision-maker),
2. Propose a new calibration to be evaluated,
3. Perform the simulations/experiments required to assess the quality of the proposed tuning,
4. Repeat until a maximum number of evaluations is reached.

In some sense, even Newton's method [100] relies on a surrogate: at each iteration, the algorithm constructs a quadratic approximation of the objective function at a point and selects the next calibration as the decision vector that minimizes (or maximizes) it. However, differently from the quadratic approximation of Newton's method, the surrogates used by response surface techniques aim to describe the objective function and the constraints functions in a global sense. For this reason, surrogate-based methods keep track of all the calibrations that have been tried instead of only the current best candidate (as it is common for more traditional optimization algorithms). The most popular surrogate models for black-box and preference-based optimization are based either on *radial basis functions* [38] or *Gaussian processes* [153], which are both suited for global function approximation. The former models are deterministic while the latter have a probabilistic interpretation. Most Bayesian optimization methods actually rely on Gaussian process approximations [22, 45, 93]. At the same time, there exist several other optimization algorithms that use radial basis function surrogates, see for example [10, 11, 54, 111, 115, 116, 156]. Most response surface techniques differ on how they handle the *exploration-exploitation dilemma* [65, 149]:

- On the one hand, it might be tempting to use the surrogate models as "proxies" for the real objective/constraints functions and simply solve an approximate version of the optimization problem (*exploitation*);
- On the other hand, the only way to improve the quality of the approximations and to guarantee the global convergence of a surrogate-based method is to evaluate calibrations that are located in those regions of the search space where less information is available (*exploration*).

Global convergence is heavily related to exploration: in some sense, *any optimization algorithm that is globally convergent must be, ultimately, exhaustive* [97]. The easiest way to explore the decision space is to sample it in a uniform fashion, by constructing a grid of points at which to measure the objective/constraints functions. The very same rationale is used by one of the most straightforward global optimization methods, namely Grid Search (GS [5]). The resolution of the grid determines the accuracy of the solution returned by GS [5]. Clearly, such method is extremely inefficient: *the number of samples scales exponentially in the number of decision variables*. Instead, the goal of surrogate-based methods is to minimize the number of samples at which to evaluate the objective/constraints functions, making the pure exploratory approach impractical. For this reason, most response surface techniques select new calibrations for evaluation by trading off exploration and exploitation. The criterion used to make the selection is called the *infill sampling criterion*. Typically, it involves solving an additional global optimization problem wherein an *acquisition function* (which trades off exploration and exploitation) is either maximized or minimized. Some widely known acquisition functions for Bayesian optimization are the probability of improvement [79] and the expected improvement [94]. In the non-Bayesian framework, many methods use an explicit trade-off between the objective function surrogate, an exploration function and a penalization term for the black-box constraints, see for example [10, 11, 116, 156].

Contributions

This book focuses on surrogate-based methods for black-box and preference-based optimization.

(1) *A first minor contribution of this book is a unified dissertation on global, black-box and preference-based optimization*, which helps us to better contextualize surrogate-based methods. Typically, these three frameworks are treated separately in the literature, due to their vastness. In this book, we highlight how global, black-box and preference-based optimization procedures are all leaves of the same tree, which aim to solve the same global optimization problem using different information on the objective function and the constraints. Furthermore, we show that the exploration-exploitation dilemma is a recurrent topic in all three frameworks.

We are mainly interested in those surrogate-based methods that rely on radial basis function approximations. In particular, we focus on four recent procedures: (i) GLobal minimum using Inverse distance weighting and Surrogate radial basis functions (GLIS [10]), which is used for black-box optimization (with no unknown constraints functions), (ii) GLISp [11], that is an extension of the latter method to the preference-based framework, and (iii/iv) C-GLIS/C-GLISp [156], which add unknown (possibly decision-maker-based) constraints to the picture. We derive several extensions of the just mentioned methods:

(2) *The second contribution of this book is the definition of the algorithms `GLIS-r` [108] and `GLISp-r` [109], which extend `GLIS` [10] and `GLISp` [11] respectively.* In particular, we address some limitations of the exploration function used by the latter methods and propose a revisited infill sampling criterion. Moreover:

- We provide a proof of convergence for `GLIS-r` [108] and `GLISp-r` [109]. Currently, no such proof is available for `GLIS` [10] and `GLISp` [11]. *To the best of our knowledge, in the preference-based optimazion literature, no other surrogate-based method has convergence guarantees.* In this book, we show, by leveraging notions of utility theory [104], that it is possible to prove the convergence of a preference-based optimization procedure as we would for any other global or black-box optimization algorithm.
- We show on several benchmark optimization problems that `GLISp-r` [109] is more robust than `GLISp` [11], i.e. the proposed method is less likely to get stuck on local minima of global optimization problems, without sacrificing its convergence speed.

(3) *The third contribution of this book is the derivation of a general surrogate-based scheme for black-box and preference-based optimization, which we refer to as generalized Metric Response Surface (`gMRS` [108])* due to its similarities to `MSRS` [116], a popular black-box optimization procedure. `gMRS` [108] is able to handle any continuous surrogate model and any proper exploration function. *Its global convergence is also addressed.* In practice, we show that `GLIS-r` [108] and `GLISp-r` [109] are none other than two implementations of the `gMRS` [108] scheme.

(4) *The fourth contribution of this book is the extension of the `GLIS-r` [108] and `GLISp-r` [109] procedures to the case when also unknown (possibly decision-maker-based) constraints are present, giving rise to `C-GLIS-r` and `C-GLISp-r`.* Differently from `C-GLIS` and `C-GLISp` [156]:

- We modify the Probabilistic Support Vector Machine classifier [15, 106], making it more suitable for black-box and preference-based optimization, and use it to estimate the probability of feasibility of a calibration of the decision vector (with respect to the unknown constraints).

- We propose an infill sampling criterion that behaves differently based on whether `C-GLIS-r` and `C-GLISp-r` have found a feasible calibration or not.

The proposed methods, `C-GLIS-r` and `C-GLISp-r`, are shown to be more efficient than `C-GLIS` and `C-GLISp` [156] on several benchmark optimization problems.

(5) *The fifth and last contribution of this book is the application of the proposed algorithms on a control systems case study.* In particular, we use `GLIS-r` [108], `GLISp-r` [109], `C-GLIS-r` and `C-GLISp-r` to calibrate the position controller of a hydraulic forming press. We rely on a simulator of the hydraulic press under study, which makes it possible to test several different optimization strategies. In the black-box framework, we define suitable performance indicators and use them as the objective/constraints functions. Instead, in the preference-based setting, the author of this book plays the role of the calibrator, who assesses the "goodness" of the controller tunings and states which are not acceptable (decision-maker-based constraint). We show that *preference-based optimization can be quite useful when only qualitative control specifications are available*. Furthermore, we highlight the *advantages of adding the unknown (possibly decision-maker-based) constraints: (i) fewer unsatisfactory calibrations are tested and (ii) the exploration of the whole decision space is limited, making it possible to find "good" controller tunings in fewer simulations*.

Book outline

The remainder of this book is organized as follows:

- **Part I** reviews global, black-box and preference-based optimization in a unified manner and can be considered as a minor contribution of this book.
  - **Chapter 1** defines the global optimization problem, presents key concepts related to the convergence of any global optimization method and reviews several general purpose algorithms, highlighting different strategies for tackling the exploration-exploitation dilemma.
  - **Chapter 2** covers the black-box optimization framework with a focus on surrogate-based methods. The most popular surrogate models are reviewed and a plethora of infill sampling criteria are examined.
  - **Chapter 3** presents the definition of the preference-based optimization problem from an utility theory perspective and shows how surrogate-based methods can be employed to solve it.
  - **Chapter 4** reviews the `GLIS` [10], `GLISp` [11], `C-GLIS` and `C-GLISp` [156] algorithms.

- **Part II** covers the theoretical contributions of this book.
  - **Chapter 5** describes the proposed extensions `GLIS-r` [108] and `GLISp-r` [109], addressing their convergence to the global minima of the black-box and preference-based optimization problems respectively. Then, the general surrogate-based scheme, `gMRS` [108], is derived. Its convergence is also proven.
  - **Chapter 6** presents algorithms `C-GLIS-r` and `C-GLISp-r`, which extend the `GLIS-r` [108] and `GLISp-r` [109] procedures to manage unknown (possibly decision-maker-based) constraints.
- **Part III** contains the empirical results achieved by the proposed procedures on several benchmark optimization problems, as well as the control systems case study.
  - **Chapter 7** reports a thorough comparison between the proposed procedures, i.e. `GLIS-r` [108], `GLISp-r` [109], `C-GLIS-r` and `C-GLISp-r`, and the original methods, namely `GLIS` [10], `GLISp` [11], `C-GLIS` and `C-GLISp` [156].
  - **Chapter 8** shows the application of the presented black-box and preference-based algorithms for the calibration of the position controller of a hydraulic forming press.
- Lastly, the appendices of this book are organized as follows.
  - **Appendix A** reviews the mathematical background for this book (order theory, metric spaces, differentiability of multivariable functions and basic optimization notions).
  - **Appendix B** reports the benchmark optimization problems used to compare the algorithms, in Chapter 7.
  - **Appendix C** describes the settings for the black-box and preference-based optimization procedures used in Part III.

Part I

Literature review and background

Chapter 1. Global optimization

This chapter and the following two present a thorough but gentle review of the optimization frameworks of interest: Global Optimization (GO), Black-Box Optimization (BBO) and Preference-Based Optimization (PBO). The main contributions of this book concern mostly the latter two frameworks. However, GO, BBO and PBO are deeply intertwined, motivating a broader examination. Furthermore, surrogate-based methods (that are the main focus of this book) are none other than a family of global optimization algorithms. *Due to the vastness of global, black-box and preference-based optimization, these topics are often treated separately in the literature. Here, we provide a unified dissertation on the three frameworks, which we consider to be a minor contribution of this book.*

Roughly speaking, an optimization algorithm is a (often iterative) procedure that exploits the information at hand on the objective function and the constraints functions to look for the solution(s) of an optimization problem. To acquire information, the algorithm must evaluate an appropriate number of samples of the decision vector. *The main difference between GO, BBO and PBO lies in how the sample evaluation process is assumed to be carried out and its resulting data*, as highlighted by the following Definition.

Definition 1.1: Sample evaluation. *In this book, we use the term sample evaluation to refer to the process of extracting information on the optimization problem at hand from a given sample (interchangeably: point, calibration, tuning) of the decision vector. Global, black-box and preference-based optimization procedures make the following assumptions on the objective function and the constraints functions that constitute the optimization problem:*

- *Global optimization: the analytical formulations of the objective function and the constraints functions are available. A sample evaluation is simply the computation of the values assumed by those functions at the given point. Derivative information can also be included if the gradients (and the Hessians) of the objective function and the constraints functions are also known, or if they can be estimated reliably (for example, through finite differentiation [100]).*
- *Black-box optimization: the objective function and, possibly, some of the constraints functions can only be measured by running computer simulations or performing real-world experiments; no analytical formulations are available. Therefore, a sample evaluation*

amounts to executing one or several (possibly time-consuming and/or expensive) simulations/experiments to measure the values of the functions of interest at a given point.

- *Preference-based optimization: the only way to evaluate samples is through the interaction with a human Decision-Maker (DM). In particular, information on the objective function comes in the form of preferences (e.g. "this calibration is better than that one"), which are obtained by asking the DM to compare a sample to one or several other tunings of the decision vector that have been previously evaluated. In turn, similarly to black-box optimization, this might require running computer simulation or performing real-world experiment, depending on how the decision-maker expresses his/her judgement. Instead, decision-maker-based constraints require asking the DM whether or not a sample is feasible or not, based on his/her (subjective) criterion. To summarize: a sample evaluation in the preference-based optimization setting amounts to querying the decision-maker in order to extract preferences and, possibly, feasibility information.*

The rest of this Chapter is organized as follows. In Section 1.1, we present the global optimization problem, whose solution is the goal of any global, black-box and preference-based optimization algorithm. We review several key concepts that are fundamental for GO, BBO and PBO. In particular, we address the existence of a global solution as well as the necessary and sufficient condition that ensures the global convergence of any algorithm. Then, in Section 1.2, we briefly present some general purpose global optimization methods, highlighting different ways to tackle the exploration-exploitation dilemma. We also give a rough taxonomy of GO methods, contextualizing response surface techniques.

1.1 The global optimization problem

As the name implies, Global Optimization (GO) is a branch of optimization tasked with finding the *global* optimizers of an objective function over some constraint set. In this book, without loss of generality, we consider the (constrained) minimization of a certain cost function. Typically, global optimization procedures assume that the cost function is *multimodal*, i.e. it has several local minimizers over the constraint set. The next Definition presents the global optimization problem.

Definition 1.2: Global optimization problem [143]. *We define the* <u>*Global Optimization Problem (GOP)*</u> *as follows:*

$$\begin{aligned} \mathcal{X}^* &= \arg \min_{\boldsymbol{x}} f(\boldsymbol{x}) \\ \textit{s.t.} \quad &\boldsymbol{x} \in \Omega, \end{aligned} \tag{1.1}$$

where:

- $\boldsymbol{x} = \begin{bmatrix} x^{(1)} & \dots & x^{(n)} \end{bmatrix}^\top \in \mathbb{R}^n, n \in \mathbb{N}$, *is the* <u>*decision vector*</u> *and* $x^{(j)}$ *is the* j*-th* <u>*decision variable,*</u>
- $f : \mathbb{R}^n \to \mathbb{R}$ *is the* <u>*cost function,*</u>
- $\Omega \subseteq \mathbb{R}^n$ *is the* <u>*constraint set*</u> *(or* <u>*feasible region/set)*</u> *of the GOP* (1.1),
- $\mathcal{X}^*$ *is the* <u>*set of global minimizers*</u> *of the GOP* (1.1). *If at least a solution exists,* $\mathcal{X}^*$ *is defined as:*

$$\mathcal{X}^* = \left\{ \boldsymbol{x}_i^* : \boldsymbol{x}_i^* \in \Omega, f\left(\boldsymbol{x}_i^*\right) = f^* = \min_{\boldsymbol{x} \in \Omega} f(\boldsymbol{x}), i = 1, \dots, N^* \right\}, \tag{1.2}$$

  otherwise $\mathcal{X}^* = \emptyset$. $N^* \in \mathbb{N} \cup \{0\}$ *is the number of global minimizers of the GOP* (1.1).

- f^* *in* (1.2) *is the* <u>*global minimum*</u> *of the GOP* (1.1).

There exist several global optimization algorithms which make different assumptions on the feasible region Ω of the GOP (1.1), such as:

- $\Omega = \mathbb{R}^n$, i.e. the GOP (1.1) is *unconstrained*;
- The GOP (1.1) is *bound (or box) constrained*, which means that:

$$\Omega = \{\boldsymbol{x} : \boldsymbol{l} \leq \boldsymbol{x} \leq \boldsymbol{u}\}, \tag{1.3}$$

  where $\boldsymbol{l}, \boldsymbol{u} \in \mathbb{R}^n, \boldsymbol{l} \leq \boldsymbol{u}$, and the inequalities are to be considered component-wise;

- Ω includes several linear and nonlinear constraints, such as:

$$\begin{aligned} \Omega = \Big\{\boldsymbol{x} : \boldsymbol{l} \leq \boldsymbol{x} \leq \boldsymbol{u}, & \qquad \text{bounds} \\ A_{ineq} \cdot \boldsymbol{x} \leq \boldsymbol{b_{ineq}}, & \qquad \text{linear inequalities} \\ A_{eq} \cdot \boldsymbol{x} = \boldsymbol{b_{eq}}, & \qquad \text{linear equalities} \end{aligned} \tag{1.4}$$

$$\left.\begin{aligned} \boldsymbol{g_{ineq}}(\boldsymbol{x}) &\leq \boldsymbol{0}_{q_3}, && \text{nonlinear inequalities} \\ \boldsymbol{g_{eq}}(\boldsymbol{x}) &= \boldsymbol{0}_{q_4}\right\}. && \text{nonlinear equalities} \end{aligned}$$

In (1.4), we differentiate between linear and nonlinear constraints. Moreover, $\boldsymbol{l}, \boldsymbol{u} \in \mathbb{R}^n, \boldsymbol{l} \leq \boldsymbol{u}$, $A_{ineq} \in \mathbb{R}^{q_1 \times n}$, $\boldsymbol{b_{ineq}} \in \mathbb{R}^{q_1}$, $A_{eq} \in \mathbb{R}^{q_2 \times n}$, $\boldsymbol{b_{eq}} \in \mathbb{R}^{q_2}$, $\boldsymbol{g_{ineq}} : \mathbb{R}^n \to \mathbb{R}^{q_3}$ and $\boldsymbol{g_{eq}} : \mathbb{R}^n \to \mathbb{R}^{q_4}$. $q_1, q_2, q_3, q_4 \in \mathbb{N} \cup \{0\}$ are the numbers of constraints belonging to each category. Notation-wise, $\boldsymbol{0}_{q_3}$ represents the q_3-dimensional zero column vector (and similarly for $\boldsymbol{0}_{q_4}$). Typically, global optimization procedures only consider inequality constraints, hence Ω in (1.4) can be concisely defined as:

$$\Omega = \left\{\boldsymbol{x} : \boldsymbol{l} \leq \boldsymbol{x} \leq \boldsymbol{u}, \boldsymbol{g}(\boldsymbol{x}) \leq \boldsymbol{0}_q\right\}, \tag{1.5}$$

where $\boldsymbol{g} : \mathbb{R}^n \to \mathbb{R}^q$, $\boldsymbol{g}(\boldsymbol{x}) = \begin{bmatrix} g^{(1)}(\boldsymbol{x}) & \dots & g^{(q)}(\boldsymbol{x}) \end{bmatrix}^\top$, $q \in \mathbb{N} \cup \{0\}$, includes both linear and nonlinear inequality constraints.

Depending on the problem at hand, the set of global minimizers of the GOP (1.1), i.e. $\mathcal{X}^*$ in (1.2), could be empty. The following Theorem gives us sufficient conditions that ensure the existence of a global minimizer.

Theorem 1.1: Extreme Value Theorem [5]

Suppose that $f : \mathbb{R}^n \to \mathbb{R}$ is a continuous function and $\Omega \subset \mathbb{R}^n$ is a compact set. Then:

$$\arg \min_{\boldsymbol{x} \in \Omega} f(\boldsymbol{x}) \neq \emptyset.$$

Often, it is sufficient to find a single point in $\mathcal{X}^*$. However, *typically, it is not possible to seek an exact solution, $\boldsymbol{x}_i^* \in \mathcal{X}^*$, using a finite number of sample evaluations*. The following Theorem gives us necessary and sufficient conditions for the asymptotic (i.e. as the number of sample evaluations goes to infinity) convergence of a global optimization method.

Theorem 1.2: Convergence of a global optimization algorithm [143]

Let $\Omega \subset \mathbb{R}^n$ be a compact set. An algorithm converges to the global minimum of every continuous function $f : \mathbb{R}^n \to \mathbb{R}$ over Ω if and only if its sequence of iterates,

$$\langle \boldsymbol{x}_i \rangle_{i \geq 1} = \langle \boldsymbol{x}_1, \boldsymbol{x}_2, \dots \rangle,$$

is dense in Ω.

In a sense, *Theorem 1.2 says that any convergent method must be ultimately (i.e. if allowed to run indefinitely) exhaustive, although the detailed exploration of the unpromising regions of Ω might be* *delayed* [97]. In practice, instead of finding an exact solution of the GOP (1.1), a more realistic goal is to seek an ϵ-optimal solution, which is defined as follows.

Definition 1.3: ϵ-optimal solution [59]. *Consider the GOP in* (1.1) *and assume that $\mathcal{X}^* \neq \emptyset$. Suppose that an algorithm* `A` *has been used to solve the optimization problem and let $\boldsymbol{x}_{\boldsymbol{best}} \in \Omega$ be its best-found candidate. Define an* *optimality accuracy* *$\epsilon \in \mathbb{R}_{>0}$. We say that* `A` *has obtained an* *ϵ-optimal solution* *for the GOP* (1.1) *if:*

$$\min_{\boldsymbol{x}_i^* \in \mathcal{X}^*} \left\| \boldsymbol{x}_{\boldsymbol{best}} - \boldsymbol{x}_i^* \right\|_2 \leq \epsilon \text{ or} \tag{1.6a}$$

$$\left| f\left(\boldsymbol{x}_{\boldsymbol{best}}\right) - f^* \right| \leq \epsilon. \tag{1.6b}$$

In (1.6a), the optimality is evaluated based on the distance between $\boldsymbol{x}_{\boldsymbol{best}}$ and its closest global minimizer in $\mathcal{X}^*$; instead, in (1.6b), only the cost function values are considered. Clearly, if $f(\boldsymbol{x})$ is Lipschitz continuous, the two definitions of ϵ-optimality are closely related. In general, *any algorithm which guarantees to find an ϵ-optimal solution in finite time must make a number of sample evaluations that is, in the worst case, exponential in the number of decision variables n* [97]. Intuitively, this bound arises from the fact that we need to evaluate a set of samples which contains points that are sufficiently near to all points in the constraint set Ω; in turn, this means that we have to construct something like a grid of points in n dimensions with a resolution determined by ϵ [85]. The number of points in such a grid scales exponentially in n. This rationale is followed by one of the simplest global optimization algorithms, namely Grid Search (`GS` [5]), which we cover in Section 1.2.2.

1.2 Global optimization algorithms

Up until now we have only described the global optimization problem in general, without referring to any particular algorithm that can be used to solve it. In practice, there exist an extensive amount of global optimization methods. On the surface, we can distinguish between *derivate-free* and *derivative-based* algorithms [118]. The former procedures do not rely on the gradient (and Hessian) of the cost function, nor those of the constraints functions, in any way. The search is carried out using only the cost function and constraints functions values obtained through sample evaluations. Also, no attempt is made to approximate the derivatives (for example, by means of finite differentiation [100]). Vice-versa for derivative-based methods, which either assume that the derivatives of the cost function (and, possibly, of the constraints functions) are available or can be estimated in some way. Before introducing a more thorough taxonomy for global optimization methods, we mention some key principles that need to be taken into consideration when defining optimization algorithms.

Remark 1.1 (Principles for constructing optimization algorithms [143])**.** *There are three key principles for the definition and the comparison of optimization algorithms:*

1. *A method that uses all available a-priori information will outperform any other algorithm that relies on less information. Some examples of a-priori information in the optimization context are:*
   - *Analytical expressions of the cost function $f(\boldsymbol{x})$ and its derivatives,*
   - *Lipschitz constant of $f(\boldsymbol{x})$ (in case the cost function is Lipschitz continuous),*
   - *Approximate location of one of the global minimizers of the GOP* (1.1).
2. *If no a-priori information is available, then an algorithm must rely only on the information brought by the evaluation of a set of samples.*
3. *Given a fixed number of sample evaluations, optimization algorithms will only differ from each other in the distribution of the tried points.*

The latter principles give us some insight on how to compare different algorithms.

Definition 1.4: Sample efficiency of optimization algorithms. *Consider two global optimization algorithms $\mathtt{A}_1$ and $\mathtt{A}_2$. Suppose that $\mathtt{A}_1$ and $\mathtt{A}_2$ are used to solve a global optimization problem* (1.1) *to a prescribed optimality accuracy $\epsilon \in \mathbb{R}_{>0}$ (as in Definition 1.3). Define $N_{\mathtt{A}_1}$ and $N_{\mathtt{A}_2}$ as the number of sample evaluations required respectively by $\mathtt{A}_1$ and $\mathtt{A}_2$ to find an ϵ-optimal solution. Then, we say that $\mathtt{A}_1$ is more* <u>*sample efficient*</u> *than $\mathtt{A}_2$ on the specific GOP and for a given ϵ if and only if*[a]

$$N_{\mathtt{A}_1} < N_{\mathtt{A}_2}.$$

[a]Clearly, this is only one of many possible ways of comparing different algorithms. Alternatively, we could assess which method shows, on average, lower execution times. We also would like to add that, in order to achieve a fair comparison, both algorithms should be initialized in a similar fashion. For example, if $\mathtt{A}_1$ and $\mathtt{A}_2$ require an initial pool of samples from which to start the search, then the same set of points should be fed to the two procedures.

Intuitively, the sample efficiency of a method must depend on the optimization problem at hand, as well as on the chosen optimality accuracy. Some GOPs (1.1) are generally much harder than others; for example, a non-convex problem with many local minimizers is definitely harder than a convex one. At the same time, some algorithms (such as `DIRECT` [67]) are able to find a promising region of Ω (i.e. one which is likely to contain a global minimizer) quite fast but struggle when performing local search. Therefore, these methods could take an excruciatingly high amount of sample evaluations to achieve an ϵ-optimal solution when ϵ is “very small”.

Notation and conventions. In this book (as in most of the optimization literature) we use the term "sample efficient" more loosely. In particular, we say that an algorithm is sample efficient if, when compared to other similar methods, it converges faster to a "good" solution of the GOP (1.1) on several benchmark global optimization problems.

The key to sample efficiency is a proper balance between two (often) conflicting objectives [97, 139, 149]:

1. Exploitation (or *local search*): the algorithm uses all the knowledge available on the optimization problem (i.e. the previously evaluated samples and, possibly, a-priori information) to select new candidate points that are likely to improve the current best solution. Exploitation often amounts to performing a local search over the most promising region of Ω found by the algorithm, with the aim of finding a local (or possibly global) solution of the GOP (1.1).

2. Exploration (or *global search*) of the whole constraint set Ω, so that no region which could possibly contain a global minimizer of the GOP (1.1) is left uncharted.

Before moving on to reviewing some popular global optimization algorithms, which differ on how they handle the exploration-exploitation dilemma, we present a taxonomy of GO methods. Due to the fact that global optimization is extremely broad, several different taxonomies have been proposed throughout the years, see for example [97, 139, 143]. In this book, we consider three macro-categories of global optimization methods:

1. Exact (or complete) methods [97] are deterministic algorithms which guarantee to find a global minimizer of the GOP (1.1) (or, rather, an ϵ-optimal solution) using a predictable amount of resources, such as sample evaluations or computation time. Typically, sufficient a-priori information on the cost function is required for such guarantee and in order to define a proper stopping criterion. Exact methods mostly rely on two principles:

   - The branching principle: the GOP (1.1) is recursively split into subproblems which are, sooner or later, easy to solve. Since, in practice, only a limited number of points can be evaluated, the performances of a pure branching scheme depend on its ability to find a good ordering of samples to be evaluated, for which premature termination has no severe effect. Some examples of exact methods which follow the branching principle are *Grid Search (`GS` [5])* (Section 1.2.2) and *DIvide a hyper-RECTangle (`DIRECT` [67])* (Section 1.2.3).

- The branch and bound principle: the GOP (1.1) is recursively split into subproblems and for each of these we compute a bound on the cost function. The bounds are then used to eliminate those subproblems which cannot lead to a point that is better than (or at least as good as) the best sample found so far. See [87, 97] for an in-depth look at branch and bound methods.

2. Direct methods [143] are all those methods which only rely on the information brought by the evaluated samples (i.e. their locations, the values of the cost function and the constraints functions, possibly including their derivatives) to carry out the search. In practice, these methods can be globally convergent but, differently from exact methods, have no way of knowing when an ϵ-optimal solution has been found since they make no assumptions on the GOP (1.1) (such as Lipschitz continuity of $f(\boldsymbol{x})$). Hence, a typical stopping criterion is a maximum number of sample evaluations/iterations or a maximum tolerated execution time. This macro-category is extremely broad, here we just mention three popular families of direct methods:

    a) Random search methods, which perform the optimization by relying on randomly generated candidate samples. These can be used as starting points for local optimization procedures. Random search methods include the well-known *multi-start methods* (Section 1.2.4)

    b) Trajectory methods; these operate similarly to a more traditional local optimization procedure, but include exploration strategies that prevent them from getting stuck on local minima. A popular trajectory method for unconstrained and bound constrained global optimization is *simulated annealing* [75], which operates as follows. The algorithm starts from a point $\boldsymbol{x}_1 \in \Omega$. At each iteration k, the procedure generates a sample, $\boldsymbol{x}_{k+1}$, in a neighborhood of its current best solution, $\boldsymbol{x}_k$, based on some probabilistic sampling rule. Then, $\boldsymbol{x}_{k+1}$ replaces $\boldsymbol{x}_k$ either if $f(\boldsymbol{x}_{k+1}) \leq f(\boldsymbol{x}_k)$ or with probability

    $$\exp\left\{-\frac{f(\boldsymbol{x}_{k+1}) - f(\boldsymbol{x}_k)}{T_k}\right\}$$

    when $f(\boldsymbol{x}_{k+1}) > f(\boldsymbol{x}_k)$ (acceptance rule). The algorithm is stopped once a maximum number of iterations is reached. Simulated annealing is inspired from the fact that the heating (annealing) and slow cooling of a metal brings it into a more uniformly crystalline state that is believed to be the state where the free energy of bulk matter takes its global minimum [97]. Similarly, the acceptance rule depends on the parameter $T_k \in \mathbb{R}_{>0}$ (called the temperature) that decreases (cools) as the iterations go on, making it less likely to accept new samples $\boldsymbol{x}_{k+1}$ that bring no improvement.

   c) Population-based methods [139] which carry out the optimization by initializing and updating a set of candidate samples (called the population). These algorithms typically follow nature-inspired heuristics, using principles taken from biological evolution, physics and social behavior. Well-known procedures that pertain to this class are the *Particle SWARM (PSWARM [72])* optimization algorithm and *evolutionary algorithms* (such as genetic algorithms) [8].

3. Surrogate-based methods (or response surface techniques)[1] [65, 149]; these algorithms iteratively build surrogate models that approximate the cost function (and, possibly, some of the constraints functions) of the GOP (1.1) and use them to drive the search for optimal solutions. Typically, response surface techniques are more computationally demanding than the aforementioned methods due to the time required for the construction of the surrogate model(s). The aim of surrogate-based methods is the minimization of the number of sample evaluations required to achieve an ϵ-optimal solution. For this reason, response surface techniques are typically applied to those GOPs (1.1) where the cost function (and, possibly, some of the constraints functions) are expensive to evaluate. Some response surface techniques are globally convergent, although it is often of secondary concern since only few sample evaluations can be performed. Surrogate-based methods are the de facto standard algorithms for black-box and preference-based optimization. For this reason, they will be covered in detail in Chapter 2 and Chapter 3.

Note that the presented taxonomy is by no means complete, it only serves us to contextualize black-box and preference-based optimization, as well as to give a brief overview of the most common families of algorithms. For a more thorough examination see [97, 139, 143].
Several methods for global optimization have been proposed in the past years. Here, we forego an exhaustive analysis and instead review a series of algorithms that pertain to different categories.

1.2.1 Relationship to local optimization

Before moving on with our dissertation on global optimization, it is important to highlight its relationship with local optimization. So far, we have seen how solving GOPs (1.1) can be particularly complex, requiring a number of sample evaluations that is, in the worst-case, exponential in the number of decision variables. The difficulties in global optimization stem mainly from the fact that [97, 143]:

[1]Surrogate-based methods are also referred to as indirect methods in [143].

1. There are generally many local minimizers of $f(\boldsymbol{x})$ over Ω but only one (a few) of them is (are) the global minimizer(s),

2. The feasible region described by Ω may be disconnected,

3. General criteria for assessing whether a point $\boldsymbol{x} \in \Omega$ is a global solution for the GOP (1.1) and not just a local minimizer are much harder to define. In particular, it is not sufficient to impose some conditions on the gradient and the Hessian of $f(\boldsymbol{x})$ (such as the ones in Appendix A.4). See for example [59, 143] for some of these criteria.

Most often, an exhaustive search is needed to guarantee that the global minimizers of the GOP (1.1) are found. However, some applications do not actually require seeking a global solution $\boldsymbol{x}_i^* \in \mathcal{X}^*$ in (1.2): a local solution $\boldsymbol{x}^+ \in \Omega$, $f\left(\boldsymbol{x}_i^*\right) \leq f\left(\boldsymbol{x}^+\right)$, might suffice.
Roughly speaking, local optimization procedures typically start from a single point, $\boldsymbol{x}_1 \in \mathbb{R}^n$, and try to improve upon it by producing a sequence of iterates $\langle \boldsymbol{x}_i \rangle_{i \geq 1}$ such that $f(\boldsymbol{x}_1) > f(\boldsymbol{x}_2) > \dots$ [100]. Typically, these methods do not keep track of all the points that have been tried but rather only of the current best candidate. Conversely, as we will see shortly, global optimization procedures often save most or all the samples that have been previously evaluated, as these provide valuable (global) information on the GOP (1.1). The next Remark further distinguishes between local and global optimization algorithms.

Remark 1.2 (Relationship between local and global optimization [59])**.** *Local optimization procedures also try to solve Problem* (1.1) *but, differently from global optimization algorithms, they are content with finding a feasible point* $\boldsymbol{x}^+ \in \Omega$ *such that*

$$f\left(\boldsymbol{x}^+\right) \leq f(\boldsymbol{x}), \quad \forall \boldsymbol{x} \in \Omega \cap \mathcal{N}(\boldsymbol{x}_1),$$

where $\boldsymbol{x}_1$ *is the starting point for the local optimization procedure and* $\mathcal{N}(\boldsymbol{x}_1)$ *is some neighborhood of* $\boldsymbol{x}_1$*. Thus,* $\boldsymbol{x}^+$ *is only a local solution of Problem* (1.1)

A particular case are convex optimization problems, for which any local solution is also a global solution of the GOP (1.1) (see Appendix A.4).
Local optimization algorithms do not need to generate a sequence of iterates that is dense in Ω *to ensure their convergence to a local minimizer*. Furthermore, there exist first-order and second-order conditions that characterize the local minimizers of Problem (1.1), see Appendix A.4. These are often used as a stopping criteria for local optimization algorithms. Strong theoretical results are also available on the convergence rates of several of these methods. In the case of unconstrained optimization problems, under some hypotheses, the steepest descent algorithm has a linear convergence

rate whereas Newton and quasi-Newton methods show quadratic and superlinear convergence rates respectively [100]. For this reason, local optimization methods can be particularly sample efficient (in a sense that they converge to local minima quite fast), even when there are several thousands of decision variables. Finally, there exist many local optimization methods which can handle nonlinear equality and inequality constraints, such as Sequential Quadratic Programming, Penalty and Barrier methods [100]. In comparison, global optimization procedures typically assume that the feasible region is composed of only simple bounds on the decision variables (i.e. with Ω as in (1.3)). However, as we will see in Section 1.2.6, penalty functions can also be used to handle nonlinear inequality constraints for global optimization procedures.

To quote [85], "global optimization is hopelessly hard while local optimization is hopelessly easy". If the GOP (1.1) to be solved is convex, then there is no reason to use global optimization procedures. However, in practice, many real-world applications require solving non-convex optimization problems, hoping to find a global solution. These include: packing problems, scheduling problems, nonlinear Least Squares problems and several engineering design problems [2, 57, 97], just to cite a few.

As a final remark, note that many global optimization procedures include local searches at the end: initially, a GO method $\mathtt{A}_1$ locates a promising region of Ω that is likely to contain a global minimizer of the GOP (1.1); then, a local optimization algorithm $\mathtt{A}_2$ looks for a more accurate solution, starting from the best point found by $\mathtt{A}_1$.

1.2.2 Grid Search

One of the most straightforward algorithms for global optimization is Grid Search (GS) [5], which can be applied to bound constrained GOPs (1.1), i.e. with Ω defined as in (1.3). We take advantage of its simplicity to give some insights on how to address the global convergence of any GO procedure.

The optimization is carried out by generating a uniformly spaced grid of points and evaluating the cost function at all the samples, returning the one which achieves the lowest value. Each decision variable $x^{(i)}, 1 \leq i \leq n$, is discretized so that it can only assume $n_g \geq 2, n_g \in \mathbb{N}$, possible values that are equidistant in the interval $\left[l^{(i)}, u^{(i)}\right]$. In total, the grid is composed of $\left(n_g\right)^n$ points, which are all possible combinations among the discrete values that each $x^{(i)}$ can assume. Therefore, the total number of sample evaluations is:

$$N = \left(n_g\right)^n . \tag{1.7}$$

Algorithm 1 describes one possible implementation of the Grid Search optimization procedure while Figure 1 shows a two-dimensional example of the samples obtained by GS [5], with $n_g = 4$ and for $\boldsymbol{l} = \boldsymbol{0}_2, \boldsymbol{u} = \boldsymbol{1}_2$.

Algorithm 1: Grid Search (GS [5])

Input: (i) Cost function $f(\boldsymbol{x})$ of the GOP (1.1); (ii) Lower bounds $\boldsymbol{l} \in \mathbb{R}^n$ and upper bounds $\boldsymbol{u} \in \mathbb{R}^n$ of the GOP (1.1); (iii) Number of discrete values for each decision variable $n_g \geq 2, n_g \in \mathbb{N}$.

Output: (i) Best cost obtained by the procedure y_{best}, (ii) Best sample obtained by the procedure $\boldsymbol{x_{best}}$.

```
1: Compute the step sizes $\delta^{(j)} = \frac{u^{(j)}-l^{(j)}}{n_g-1}, \forall j = 1, \dots, n$
2: Generate the grid of samples:
       $\mathcal{X} = \left\{\boldsymbol{x}_i : x_i^{(j)} = l^{(j)} + k \cdot \delta^{(j)}, \forall j = 1, \dots, n, \forall k = 0, \dots, n_g - 1\right\}$
3: Evaluate all the points in $\mathcal{X}$, obtaining $\mathcal{Y} = \{y_i : f(\boldsymbol{x}_i), \boldsymbol{x}_i \in \mathcal{X}\}$
4: Select the best solution:
       $y_{best} = \min_{y_i \in \mathcal{Y}} y_i$
       $\boldsymbol{x_{best}} = \arg\min_{\boldsymbol{x}_i \in \mathcal{X}} f(\boldsymbol{x}_i)$
```

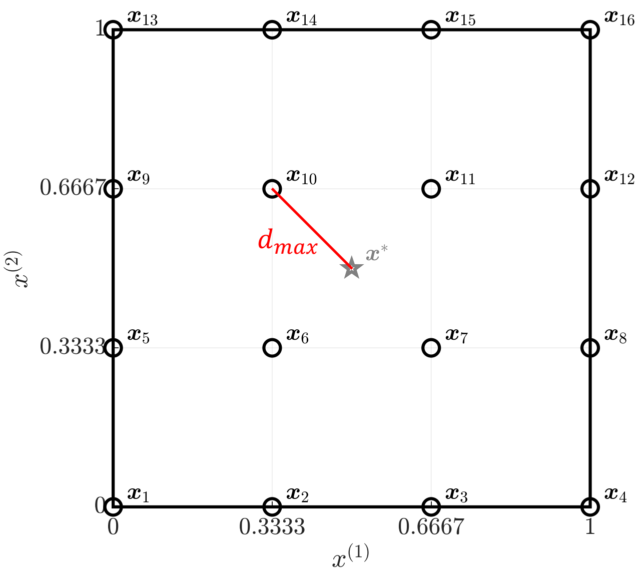


Figure 1: Two-dimensional example of the samples (black circles) obtained by the GS [5] procedure. Each decision variable has been discretized so that it can assume only $n_g = 4$ values. The bounds are $\boldsymbol{l} = \boldsymbol{0}_2$ and $\boldsymbol{u} = \boldsymbol{1}_2$. $\boldsymbol{x}^*$, denoted using a grey star, represents a possible global minimizer of the GOP (1.1) in the worst case. d_{max} (in red) is the distance from $\boldsymbol{x}^*$ to the closest point obtained by GS [5].

The convergence of the GS [5] algorithm is addressed by the following Theorem.

Theorem 1.3: Convergence of GS [5]

Let $\Omega \subset \mathbb{R}^n$ be defined as in (1.3) and $f : \mathbb{R}^n \to \mathbb{R}$ be a continuous function. Then, provided that $n_g \to \infty$, Algorithm 1 converges to the global minimum of the GOP (1.1).

Proof. Note that Ω in (1.3) is a compact subset of $\mathbb{R}^n$ (see Appendix A.2, Lemma A.2). $f(\boldsymbol{x})$ is assumed to be continuous and therefore, due to the Extreme Value Theorem 1.1, the considered GOP (1.1) admits a solution. To prove Theorem 1.3, we need to verify that the sequence of iterates $\langle \boldsymbol{x}_i \rangle_{i \geq 1}$ generated by Algorithm 1 is dense in Ω (see Theorem 1.2). To do so, consider the set of samples $\mathcal{X}$ computed by the GS [5] procedure, which contains all the elements of $\langle \boldsymbol{x}_i \rangle_{i \geq 1}$. In practice, Algorithm

1 partitions Ω into several disjoint boxes, whose vertices are the points in $\mathcal{X}$, with sides of length:

$$\frac{u^{(i)} - l^{(i)}}{n_g - 1}, \quad i = 1, \dots, n.$$

Therefore, the maximum distance $d_{max} \in \mathbb{R}_{\geq 0}$ between any point in Ω and the samples in $\mathcal{X}$ is actually the distance between the center of any of the boxes into which Ω is partitioned and its vertices (see Figure 1):

$$\begin{aligned} d_{max} &= \max_{\boldsymbol{x} \in \Omega} \min_{\boldsymbol{x}_i \in \mathcal{X}} \|\boldsymbol{x} - \boldsymbol{x}_i\|_2 \\ &= \frac{\|\boldsymbol{u} - \boldsymbol{l}\|_2}{2 \cdot \left(n_g - 1\right)}. \end{aligned} \tag{1.8}$$

We can immediately see that $d_{max} \longrightarrow 0$ for $n_g \longrightarrow \infty$. Hence, given any $\boldsymbol{x} \in \Omega$, `GS` [5] will eventually sample a point within an arbitrary distance $\epsilon \in \mathbb{R}_{>0}$ of $\boldsymbol{x}$. We can conclude that $\mathcal{X}$ is dense in Ω (see Appendix A.2, Definition A.16) and thus, by Theorem 1.2, Algorithm 1 converges to the global minimum of the GOP (1.1). □

Note that *a convergent global optimization algorithm is not necessarily efficient*. Theorem 1.3 only ensures that, if the resolution of the grid is infinitely small, then `GS` [5] finds the exact global minimum of the GOP (1.1). We can also compute the minimum number of sample evaluations required by Grid Search to find an ϵ-optimal solution (in the worst case).

Lemma 1.1. *Let $\Omega \subset \mathbb{R}^n$ be defined as in* (1.3) *and $f : \mathbb{R}^n \longrightarrow \mathbb{R}$ be a continuous function. Given an optimality accuracy $\epsilon \in \mathbb{R}_{>0}$, Algorithm 1 requires at least*

$$N \geq \left[\frac{\|\boldsymbol{u} - \boldsymbol{l}\|_2}{2 \cdot \epsilon} + 1\right]^n \tag{1.9}$$

sample evaluations to achieve an ϵ-optimal solution as defined in (1.6a)*, in the worst case.*

Proof. In the worst case, the global minimizers of the GOP (1.1) are located at the centers of the boxes into which Ω is partitioned by Algorithm 1. Consider $\boldsymbol{x}_i^*$ to be one one of such global minimizers. Then, the distance between $\boldsymbol{x}_i^*$ and its closest point in $\mathcal{X}$ is d_{max} in (1.8). Consider the notion of ϵ-optimality as defined in (1.6a). We require that:

$$\min_{\boldsymbol{x}_i \in \mathcal{X}} \left\|\boldsymbol{x}_i^* - \boldsymbol{x}_i\right\|_2 = d_{max} \leq \epsilon.$$

We can compute a worst case lower bound on n_g by substituting to d_{max} its expression in (1.8):

$$\begin{aligned} \frac{\|\boldsymbol{u}-\boldsymbol{l}\|_2}{2\cdot\left(n_g-1\right)} &\leq \epsilon \\ \frac{\|\boldsymbol{u}-\boldsymbol{l}\|_2}{2\cdot\epsilon} &\leq \left(n_g-1\right) \\ n_g &\geq \frac{\|\boldsymbol{u}-\boldsymbol{l}\|_2}{2\cdot\epsilon}+1. \end{aligned}$$

Recall that the number of sample evaluations performed by GS [5] for a certain n_g is $N = \left(n_g\right)^n$. Thus, Algorithm 1 requires at least

$$N \geq \left[\frac{\|\boldsymbol{u}-\boldsymbol{l}\|_2}{2\cdot\epsilon}+1\right]^n$$

sample evaluations to achieve an ϵ-optimal solution (as defined in (1.6a)). □

If the cost function of the GOP (1.1) is Lipschitz continuous (see Appendix A.2.4), then we can also compute the minimum number of sample evaluations required to achieve a certain optimality accuracy $\epsilon \in \mathbb{R}_{>0}$ for $f(\boldsymbol{x})$ as in (1.6b).

Lemma 1.2. *Let* $\Omega \subset \mathbb{R}^n$ *be defined as in* (1.3) *and* $f : \mathbb{R}^n \to \mathbb{R}$ *be a Lipschitz continuous function with Lipschitz constant* $C \in \mathbb{R}_{>0}$. *Given an optimality accuracy* $\epsilon \in \mathbb{R}_{>0}$, *Algorithm 1 requires at least*

$$N \geq \left[C\cdot\frac{\|\boldsymbol{u}-\boldsymbol{l}\|_2}{2\cdot\epsilon}+1\right]^n \tag{1.10}$$

sample evaluations to achieve an ϵ*-optimal solution as defined in* (1.6b)*, in the worst case.*

Proof. From (1.6b), we require that $|y_{best} - f^*| \leq \epsilon$. As we have previously seen in Lemma 1.1, in the worst case the global minimizers are situated at the centers of the boxes into which Ω is partitioned. Thus, since $f(\boldsymbol{x})$ is Lipschitz continuous, we have:

$$|y_{best} - f^*| \leq C\cdot d_{max} \leq \epsilon.$$

We can compute a worst case lower bound on n_g by substituting to d_{max} its expression in (1.8):

$$\begin{aligned} C\cdot\frac{\|\boldsymbol{u}-\boldsymbol{l}\|_2}{2\cdot\left(n_g-1\right)} &\leq \epsilon \\ n_g &\geq C\cdot\frac{\|\boldsymbol{u}-\boldsymbol{l}\|_2}{2\cdot\epsilon}+1. \end{aligned}$$

Thus, Algorithm 1 requires at least

$$N \geq \left[C\cdot\frac{\|\boldsymbol{u}-\boldsymbol{l}\|_2}{2\cdot\epsilon}+1\right]^n$$

sample evaluations to achieve an ϵ-optimal solution (as defined in (1.6b)). □

The next Remark gives us more general expressions for N in (1.9) and in (1.10) that do not depend on the bounds $\boldsymbol{l}, \boldsymbol{u}$.

Remark 1.3. *Typically, when dealing with bound constrained GOPs* (1.1)*, the decision variables are rescaled so that the feasible region becomes the n-dimensional unit hypercube, i.e.* $\boldsymbol{l} = \mathbf{0}_n$ *and* $\boldsymbol{u} = \mathbf{1}_n$*. Then, the worst case lower bound provided by Lemma 1.1 becomes:*

$$N \geq \left[\frac{\sqrt{n}}{2 \cdot \epsilon} + 1\right]^n, \tag{1.11}$$

while the one of Lemma 1.2 results in:

$$N \geq \left[\tilde{C} \cdot \frac{\sqrt{n}}{2 \cdot \epsilon} + 1\right]^n. \tag{1.12}$$

$\tilde{C}$ *is a properly rescaled Lipschitz constant for* $f(\boldsymbol{x})$*, which takes into account that the decision vector has been rescaled.*

Clearly Lemma 1.1 and Lemma 1.2 highlight how GS [5] can be extremely inefficient, even for low-dimensional problems. Furthermore, (1.9) and (1.10) show how the problem of finding the global solutions of an optimization problem is affected by the curse of dimensionality. In particular, *the number of sample evaluations required to obtain an ϵ-optimal solution (in the worst case) grows exponentially with the number of decision variables n.*

One of the main shortcomings of Algorithm 1 is that the cost function values achieved by the samples in $\mathcal{X}$ are not used in any way to help the search (in practice, Algorithm 1 only performs exploration and no exploitation). Moreover, the samples in $\mathcal{X}$ are not evaluated in any particular order. Clearly, if GS [5] were to somehow delay the evaluation of those points that are less likely to offer an improvement, its sample efficiency would increase. Nonetheless, Algorithm 1 is often used to tune the hyper-parameters of many machine learning methods [58], manly because it is easy to parallelize. In the next Section, we show a more refined algorithm that still relies on partitioning Ω into smaller boxes.

1.2.3 The DIRECT algorithm

We have seen that Grid Search can be extremely inefficient due to the fact that it evaluates all the points in a uniformly spaced grid, regardless of their cost function values. A much better alternative to GS [5] is the DIvide a hyper-RECTangle (DIRECT) [67] algorithm, which too is used for bound constrained GOPs (1.1). Instead of dividing Ω into several boxes with the same volume and evaluating the cost function at each of their vertices, DIRECT [67] sequentially partitions Ω into smaller hyper-rectangles based on the values of $f(\boldsymbol{x})$ measured at their center points.

Algorithm 2 summarizes each step of the procedure. At first, the feasible region of the GOP (1.1) is normalized so that it becomes the n-dimensional unit hypercube. At each iteration, the algorithm considers a partition of Ω:

$$\Omega_p = \left\{\Omega^{(1)}, \Omega^{(2)}, \dots\right\}, \tag{1.13}$$

i.e. such that $\Omega^{(i)} \cap \Omega^{(j)} = \emptyset, \forall i \neq j, \bigcup_{\Omega^{(i)} \in \Omega_p} \Omega^{(i)} = \Omega$; $\Omega^{(i)} \in \Omega_p$ are all boxes of (possibly) different volumes. Then, DIRECT [67] selects a set of potentially optimal boxes $\Omega_o \subseteq \Omega_p$ defined as follows. Let $\boldsymbol{x}_i$ be the center point of the i-th hyper-rectangle $\Omega^{(i)} \in \Omega_p$ and $d_i \in \mathbb{R}_{>0}$ denote the distance from $\boldsymbol{x}_i$ to the vertices of the box. Also, let $\tau \in \mathbb{R}_{>0}$ be a positive constant. A hyper-rectangle $\Omega^{(j)} \in \Omega_p$ is said to be *potentially optimal* if there exists some rate-of-change constant $\tilde{C} \in \mathbb{R}_{>0}$ such that:

$$f\left(\boldsymbol{x}_j\right) - \tilde{C} \cdot d_j \leq f\left(\boldsymbol{x}_i\right) - \tilde{C} \cdot d_i, \quad \forall i = 1, \dots, \left|\Omega_p\right|, \tag{1.14a}$$

$$f\left(\boldsymbol{x}_j\right) - \tilde{C} \cdot d_j \leq y_{best}(k) - \tau \cdot \left|y_{best}(k)\right|, \tag{1.14b}$$

where $y_{best}(k)$ is the value of the cost function achieved by the best candidate found so far (i.e. at iteration k) by the procedure. The class of potentially optimal boxes, $\Omega_o = \left\{\Omega_o^{(1)}, \Omega_o^{(2)}, \dots\right\}, \Omega_o \subseteq \Omega_p$, is composed of all those sets for which the conditions in (1.14) hold. Note that the expression $f\left(\boldsymbol{x}_j\right) - \tilde{C} \cdot d_j$ in (1.14a) can be seen as a trade-off between the cost achieved by the center point of the j-th box (exploitation) and its volume (exploration). Thus, potentially optimal hyper-rectangles are those which achieve a good balance between these two objectives. The second condition in (1.14b) prevents DIRECT [67] from performing unnecessarily accurate local searches, which result in negligible improvements. Each potentially optimal box is then partitioned into smaller hyper-rectangles, as described in Algorithm 2, and Ω_p in (1.13) is updated accordingly. The procedure is stopped once a maximum number of iterations, $k_{max} \in \mathbb{N}$, is reached.

Figure 2 depicts an application of DIRECT [67] to a two-dimensional problem. At first, Ω_p in (1.13) contains only the normalized feasible region Ω, i.e. $\Omega_p = \left\{\Omega^{(1)}\right\}$ with $\Omega^{(1)} = \Omega$. In this case, there is only one potentially optimal hyper-rectangle: $\Omega^{(1)}$. Four samples are selected for evaluation as described in Algorithm 2; then, $\Omega^{(1)}$ is trisected along $x^{(1)}$ and $x^{(2)}$. Ω_p is updated and now contains four different hyper-rectangles of different volumes. The same process is iterated two other times. In particular, at the second and third iteration there are, respectively, one and two potentially optimal hyper-rectangles.

The global convergence of DIRECT has been addressed in [67], as reported in the following Theorem.

Algorithm 2: DIvide a hyper-RECTangle (`DIRECT` [67])

Input: (i) Cost function $f(\boldsymbol{x})$ of the GOP (1.1); (ii) Lower bounds $\boldsymbol{l} \in \mathbb{R}^n$ and upper bounds $\boldsymbol{u} \in \mathbb{R}^n$ of the GOP (1.1); (iii) Desired relative accuracy $\tau \in \mathbb{R}_{>0}$; (iv) Maximum number of iterations $k_{max} \in \mathbb{N}$.

Output: (i) Best cost obtained by the procedure $y_{best}(k_{max})$; (ii) Best sample obtained by the procedure $\boldsymbol{x_{best}}(k_{max})$.

```
 1: Normalize the feasible region $\Omega$ to the $n$-dimensional unit hypercube
 2: Initialize the class containing all the sets into which $\Omega$ will be partitioned:
        $\Omega_p = \left\{\Omega^{(1)}\right\}, \quad \Omega^{(1)} = \Omega$
 3: Let $\boldsymbol{x}_1$ be the center of $\Omega^{(1)}$ and set $\mathcal{X} = \{\boldsymbol{x}_1\}$
 4: Evaluate the cost function at $\boldsymbol{x}_1$, obtaining $y_1 = f(\boldsymbol{x}_1)$. Also set $\mathcal{Y} = \{y_1\}$
 5: Initialize the best candidate: $\boldsymbol{x_{best}}(0) = \boldsymbol{x}_1$ and $y_{best}(0) = y_1$
 6: for $k = 1, \dots, k_{max}$ do
 7:     Identify the set of potentially optimal boxes as in (1.14):
            $\Omega_o = \left\{\Omega_o^{(1)}, \Omega_o^{(2)}, \dots\right\}, \quad \Omega_o \subseteq \Omega_p$
 8:     for all $\Omega_o^{(j)} \in \Omega_o$ do
 9:         Divide hyper-rectangle $\Omega_o^{(j)}$ with center $\boldsymbol{x}_j$ as follows:
            a: Identify the set $\mathcal{I} \subseteq \{1, \dots, n\}$ of dimensions with the maximum side length $\delta_{max}$
            b: Set $\delta = \frac{\delta_{max}}{3}$
            c: Select the new $2 \cdot |\mathcal{I}|$ candidate samples as $\boldsymbol{x}_j \pm \delta \cdot \boldsymbol{e}_i, \forall i \in \mathcal{I}$, and add them to $\mathcal{X}$
            d: Evaluate the cost function at all newly found points, $f\left(\boldsymbol{x}_j \pm \delta \cdot \boldsymbol{e}_i\right), \forall i \in \mathcal{I}$, and update $\mathcal{Y}$
            accordingly
            e: Divide the box $\Omega_o^{(j)}$ by trisecting along the dimensions in $\mathcal{I}$, from the one with the lowest
            $\min\left\{f\left(\boldsymbol{x}_j + \delta \cdot \boldsymbol{e}_i\right), f\left(\boldsymbol{x}_j - \delta \cdot \boldsymbol{e}_i\right)\right\}$ to the $i$-th with the highest
            f: Replace $\Omega_o^{(j)}$ in $\Omega_p$ with the sets that constitute its partition
10:         Update $y_{best}(k)$ and $\boldsymbol{x_{best}}(k)$ accordingly
```

Theorem 1.4: Convergence of `DIRECT` [67]

Let $\Omega \subset \mathbb{R}^n$ be defined as in (1.3) *and $f : \mathbb{R}^n \rightarrow \mathbb{R}$ be a continuous function. Then, provided that $k_{max} \rightarrow \infty$, Algorithm 2 converges to the global minimum of the GOP* (1.1).

The proof is similar to that of the convergence theorem for `GS` [5] (Theorem 1.3), see [67].

Due to its effectiveness and simplicity, `DIRECT` [67] has received much attention since its release. The review in [66] sums up many of the extensions of the method that have been developed in the past two decades. Notably, the authors of [34] have proposed an extension of `DIRECT` [67] that is able to handle a GOP (1.1) with any type of constraints, i.e. with Ω defined as in (1.4).

1.2.4 Multi-start methods

In Section 1.2.1 we have seen how local optimization procedures are quite efficient in finding local solutions of the GOP (1.1). A simple way to adapt these methods for global optimization is to randomly sample the feasible region Ω using a uniform distribution, obtaining a set of $N_{init} \in \mathbb{N}$ initial samples, and then start a local optimization algorithm from each of these points. This paradigm is followed

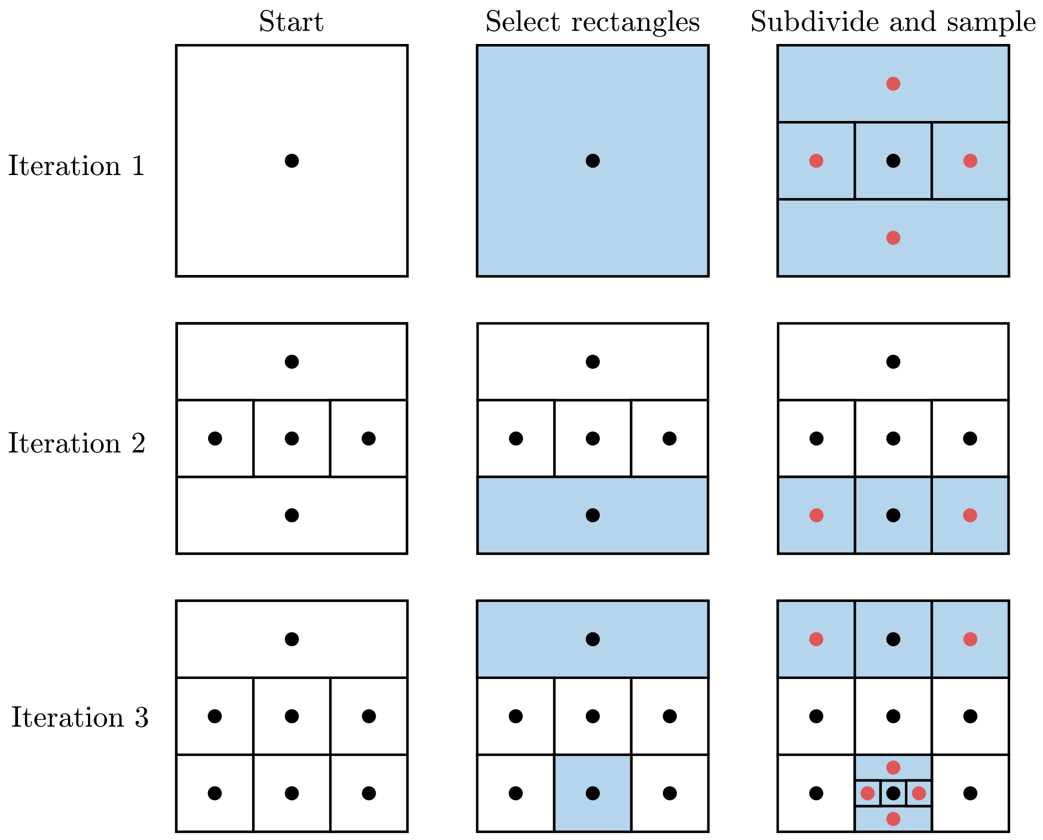


Figure 2: Two-dimensional example of the first three iterations of the `DIRECT` [67] algorithm. The black dots are the previously evaluated samples. The blue hyper-rectangles are the potentially optimal ones. Finally, the red points are those generated at Step 9 of Algorithm 2. This Figure is taken from [66].

by multi-start methods [59, 90] and described in Algorithm 3. The main advantages of multi-start methods are:

1. They can be parallelized, i.e. multiple local optimization procedures can be run simultaneously,
2. Nonlinear constraints are easily handled: we only need to select a suitable local optimization algorithm.

However, as we will see shortly, multi-start methods can be very sample inefficient. In any case, at least asymptotically, it is possible to prove their convergence (in probability) to the global minimum of the GOP (1.1).

Theorem 1.5: Convergence of a multi-start method [59]

Let Ω be a compact subset of $\mathbb{R}^n$ and $f : \mathbb{R}^n \to \mathbb{R}$ be a continuous function. Then, provided that $N_{init} \to \infty$, Algorithm 3 converges to the global minimum of the GOP (1.1) *with probability one.*

Intuitively, as the number of samples N_{init} produced by randomly sampling Ω increases, it becomes more likely to find a point $\boldsymbol{x}_j$ within a neighborhood of a global minimizer $\boldsymbol{x}_i^*$ of the GOP (1.1).

Hence, running a local optimization procedure from $\boldsymbol{x}_j$ will yield $\boldsymbol{x}_i^*$. See [136] for an in-depth look at some convergence results for random search methods.

Multi-start methods can be sample inefficient due to the fact that several runs of the local optimization procedure (started from different points) can lead to the same local minimizer of the GOP (1.1), as highlighted by the following Definition.

Definition 1.5: Basin of attraction [81, 143]. *Let $\mathbf{x}^+ \in \Omega$ be a local minimizer of the GOP* (1.1) *and* A *be a local optimization algorithm. We define the basin of attraction $\mathcal{BA}(\boldsymbol{x}^+; \texttt{A})$ as a subset of Ω such that* A *converges to $\boldsymbol{x}^+$ for any starting point $\boldsymbol{x}_1 \in \mathcal{BA}(\boldsymbol{x}^+; \texttt{A})$.*

Roughly speaking, a basin of attraction for an optimization algorithm is the set of initial samples leading to the same local minimizer. Therefore, if the random sampling phase of a multi-start method produces several points that belong to the same basin of attraction, some of the search effort is wasted. Ideally, we would like to have exactly one point inside each basin of attraction associated to all the local minimizers $\boldsymbol{x}^+$ of the GOP (1.1).

One way to make multi-start methods more efficient is to use a clustering scheme to limit the number local optimization procedures which lead to the same solution. A popular algorithm that follows such rationale is *Density Clustering* [59].

Algorithm 3: Multi-start method [59]

Input: (i) Cost function $f(\boldsymbol{x})$ of the GOP (1.1); (ii) Constraint set Ω of the GOP (1.1); (iii) Number of initial samples $N_{init} \in \mathbb{N}$ from which to start the local optimization procedure; (iv) Local optimization algorithm A.

Output: (i) Best cost obtained by the procedure y_{best}; (ii) Best sample obtained by the procedure $\boldsymbol{x_{best}}$.

```
1: Initialize best value for the cost function: $y_{best} = +\infty$
2: Generate the set of initial samples $\mathcal{X}_{init}$, $|\mathcal{X}_{init}| = N_{init}$, by randomly sampling $\Omega$ using a uniform distribution
3: for all $\boldsymbol{x}_i \in \mathcal{X}_{init}$ do
4:     Run algorithm A, starting from $\boldsymbol{x}_i$, to obtain a solution $\tilde{\boldsymbol{x}}$ with corresponding cost $\tilde{y}$
5:     if $\tilde{y} < y_{best}$ then
6:         Update the best candidate sample: $y_{best} = \tilde{y}, \boldsymbol{x_{best}} = \tilde{\boldsymbol{x}}$
```

1.2.5 Particle SWARM optimization

Particle Swarm optimization (PSWARM) is a population-based global optimization scheme that has been first proposed in [72]. Its derivation is inspired by the social behavior of bird flocks and fish schools. PSWARM [72] can be applied to bound constrained global optimization problems (hence, with Ω as in (1.3)). Here, we briefly summarize the dissertation in [144] to explain each step of the procedure. The PSWARM [72] algorithm maintains a population of particles, where each particle represents a potential solution of the GOP (1.1). The *population* is composed of $N_{pop} \in \mathbb{N}$ elements and is updated at each iteration k of the optimization procedure. We denote the population at the k-th iteration as:

$$\mathcal{P}(k) = \left\{p_i(k) : p_i(k) \in \mathbb{R}^n \times \mathbb{R}^n \times \mathbb{R}^n, i = 1, \ldots, N_{pop}\right\}. \tag{1.15}$$

A *particle* $p_i(k)$ is composed of three characteristics:

1. The current position of the particle, $\boldsymbol{x} \in \mathbb{R}^n$,
2. The current velocity of the particle, $\boldsymbol{v} \in \mathbb{R}^n$,
3. The best position achieved by the particle, $\boldsymbol{b} \in \mathbb{R}^n$.

For the remainder of this Section, we will use an object-oriented programming notation to reference each characteristic. In particular, $p_i(k) \rightarrow \boldsymbol{x}$ represents the position of the i-th particle at the k-th iteration; similarly for the other characteristics.
The best position of each particle is updated as follows:

$$p_i(k) \rightarrow \boldsymbol{b} = \begin{cases} p_i(k-1) \rightarrow \boldsymbol{b} & \text{if } f(p_i(k) \rightarrow \boldsymbol{x}) \geq f(p_i(k-1) \rightarrow \boldsymbol{b}) \\ p_i(k) \rightarrow \boldsymbol{x} & \text{if } f(p_i(k) \rightarrow \boldsymbol{x}) < f(p_i(k-1) \rightarrow \boldsymbol{b}) \end{cases}. \tag{1.16}$$

Therefore, the best candidate sample found by `PSWARM` [72] at the k-th iteration is:

$$\boldsymbol{x_{best}}(k) = \arg \min_{\boldsymbol{b}_i \in \{\boldsymbol{b}_i : \boldsymbol{b}_i = p_i(k) \rightarrow \boldsymbol{b}, p_i(k) \in \mathcal{P}(k)\}} f(\boldsymbol{b}_i) \tag{1.17}$$

and, similarly, the best cost is:

$$y_{best}(k) = f(\boldsymbol{x_{best}}(k)). \tag{1.18}$$

The velocity of each particle is updated as follows[2]:

$$\begin{aligned} p_i(k) \rightarrow v^{(j)} = p_i(k-1) \rightarrow v^{(j)} + & \\ + \delta_1 \cdot r_1^{(j)}(k) \cdot \left[p_i(k-1) \rightarrow b^{(j)} - p_i(k-1) \rightarrow x^{(j)}\right] + & \\ + \delta_2 \cdot r_2^{(j)}(k) \cdot \left[x_{best}^{(j)}(k-1) - p_i(k-1) \rightarrow x^{(j)}\right], & \end{aligned} \tag{1.19}$$

where $\delta_1, \delta_2 \in (0, 2]$ are two user-defined constants (called the *acceleration coefficients*) and $\boldsymbol{r}_1(k), \boldsymbol{r}_2(k)$ are two vectors of random variables drawn from the uniform distribution, $\boldsymbol{r}_1(k), \boldsymbol{r}_2(k) \overset{i.i.d.}{\sim} \boldsymbol{\mathcal{U}}(\boldsymbol{0}_n, \boldsymbol{1}_n)$, that are generated at each iteration. The update is performed for each dimension $j = 1, \ldots, n$. The acceleration coefficients in (1.19) trade-off the maximum step size towards the best position for the

[2]This is only one possible way to update the velocity of each particle. An alternative would be to keep track of a subset of particles for each $p_i(k)$ and from which the best one is selected (i.e. the local best candidate sample for $p_i(k)$). Then, in (1.19), the velocity would be updated using the local best candidate for the particle instead of $\boldsymbol{x_{best}}(k)$ in (1.17). This rationale is referred to as `lbest` update whereas the one in (1.19) is called `gbest` update [37].

i-th particle, $p_i(k) \rightarrow \boldsymbol{b}$, and the best candidate sample (among all particles), $\boldsymbol{x_{best}}(k)$, respectively. Finally, the position of each particle is updated as:

$$p_i(k) \rightarrow \boldsymbol{x} = p_i(k-1) \rightarrow \boldsymbol{x} + p_i(k) \rightarrow \boldsymbol{v}. \tag{1.20}$$

Note that, in order to satisfy the bound constraints in Ω (1.3), the values of $p_i(k) \rightarrow \boldsymbol{x}$ in (1.20) are clamped so that they are between $\boldsymbol{l}$ and $\boldsymbol{u}$. Similarly for the velocities computed in (1.19), although (often) more restrictive bounds are used: $\boldsymbol{l_v} \leq p_i(k) \rightarrow \boldsymbol{v} \leq \boldsymbol{u_v}$ such that $\boldsymbol{l_v}, \boldsymbol{u_v} \in \mathbb{R}^n$ and $\boldsymbol{l_v} \geq \boldsymbol{l}, \boldsymbol{u_v} \leq \boldsymbol{u}$.

Algorithm 4 highlights each step of the PSWARM [72] procedure. Several different extensions have been proposed in the past two decades. The dissertation in [144] provides a detailed review on many results for the method. We remark that *the original formulation of the PSWARM [72] procedure (Algorithm 4) is not globally convergent*, although it can be made so by integrating it with some techniques "borrowed" from random search methods. Notably, the authors of [146] merge PSWARM [72] with the pattern search framework [118] (a popular family of derivative-free optimization methods). Although the method proposed in [146] is only guaranteed to find a stationary point for the GOP (using a finite number of iterations), empirical results show that it is a highly competitive alternative to many global optimization algorithms, such as DIRECT [67].

1.2.6 Handling general constraints for global optimization

In the previous Sections, we have seen that algorithms such as GS [5], DIRECT [67] and PSWARM [72] can only handle bound constrained GOPs (1.1). That is often the case for global optimization methods. In practice, however, many optimization problems include linear and nonlinear inequality constraints (i.e. Ω is defined as in (1.5)). One way to manage those constraints that cannot be handled explicitly is to add a <u>penalty function</u> $\rho : \mathbb{R}^n \rightarrow \mathbb{R}_{\geq 0}$ to the cost function, which discourages their violation. Commonly, the *quadratic penalty function* [100] is used:

$$\rho(\boldsymbol{x}) = \sum_{j=1}^{q} \left(\max\left\{0, g^{(j)}(\boldsymbol{x})\right\}\right)^2. \tag{1.21}$$

Then, the cost function of the GOP (1.1) becomes:

$$f_\rho(\boldsymbol{x}) = f(\boldsymbol{x}) + \delta_\rho \cdot \rho(\boldsymbol{x}), \tag{1.22}$$

Algorithm 4: Particle SWARM (PSWARM [72]) optimization

Input: (i) Cost function $f(\boldsymbol{x})$ of the GOP (1.1); (ii) Lower bounds $\boldsymbol{l} \in \mathbb{R}^n$ and upper bounds $\boldsymbol{u} \in \mathbb{R}^n$ of the GOP (1.1); (iii) Number of particles $N_{pop} \in \mathbb{N}$ which constitute the population; (iv) Acceleration coefficients $\delta_1, \delta_2 \in (0, 2]$; (v) Maximum number of iterations $k_{max} \in \mathbb{N}$.

Output: (i) Best cost obtained by the procedure $y_{best}(k_{max})$, (ii) Best sample obtained by the procedure $\boldsymbol{x_{best}}(k_{max})$.

```
1: Initialize the iteration counter to $k = 1$
2: Initialize the population $\mathcal{P}(k)$ as follows:
   a: Generate the positions of each particle by drawing from the uniform random distribution on the intervals $\left[l^{(i)}, u^{(i)}\right]$:

      $p_i(k) \to x^{(j)} \sim \mathcal{U}\left(l^{(i)}, u^{(i)}\right), \quad \forall i = 1, \dots, N_{pop}, \forall j = 1, \dots, n$

   b: Generate the velocities of each particle by drawing from the uniform random distribution on the intervals $\left[l_v^{(i)}, u_v^{(i)}\right]$:

      $p_i(k) \to v^{(j)} \sim \mathcal{U}\left(l_v^{(i)}, u_v^{(i)}\right), \quad \forall i = 1, \dots, N_{pop}, \forall j = 1, \dots, n$

   c: Set $p_i(k) \to \boldsymbol{b} = p_i(k) \to \boldsymbol{x}, \forall i = 1, \dots, N_{pop}$
3: Initialize the best candidate sample $\boldsymbol{x_{best}}(k)$ and cost $y_{best}(k)$ as in (1.17) and (1.18)
4: for $k = 2, \dots, k_{max}$ do
5:     for all $p_i \in \mathcal{P}(k)$ do
6:         Update the velocity of each particle as in (1.19)
7:         Update the position of each particles as in (1.20)
8:         Update the best position of each particle as in (1.16)
9:     Update the best candidate sample $\boldsymbol{x_{best}}(k)$ and cost $y_{best}(k)$ as in (1.17) and (1.18)
```

where $\delta_\rho \in \mathbb{R}_{>0}$ is the penalty parameter. Thus, the penalized global optimization problem amounts to:

$$\begin{aligned} &\arg\min_{\boldsymbol{x}} f_\rho(\boldsymbol{x}) \\ &\text{s.t.} \quad \boldsymbol{l} \leq \boldsymbol{x} \leq \boldsymbol{u}. \end{aligned} \tag{1.23}$$

Typically, $\delta_\rho \gg 1$ in (1.22) so that constraints violations are heavily penalized.

Note that the local and global solutions of Problem (1.23) need not coincide with those of the GOP (1.1). To fix this, the penalized global optimization problem should be solved multiple times with progressively higher values of the penalty parameter δ_ρ. More formally, at each iteration k, we use a global optimization algorithm A to solve Problem (1.23). Let $\boldsymbol{x_{best_\rho}}(k)$[3] be the solution of Problem (1.23) found by A, and $\delta_\rho(k)$ be the penalty parameter at the k-th iteration. Then, provided that A is globally convergent and $\delta_\rho(k) \to \infty$ for $k \to \infty$, $\boldsymbol{x_{best_\rho}}(k) \to \boldsymbol{x}_i^*$, where $\boldsymbol{x}_i^* \in \mathcal{X}^*$ in (1.2) (see [100] for a formal proof).

[3]If A is a local optimization procedure, then $\boldsymbol{x_{best_\rho}}(k)$ is typically used as a starting point for A at the $(k+1)$-th iteration.

Clearly, solving multiple global optimization problems (for different values of δ_ρ) can be quite time-consuming. Instead, it is common to follow a two step approach: (i) run a global optimization procedure $\mathtt{A}_1$ to solve Problem (1.23) with a "sufficiently high" δ_ρ and then (ii) use a local optimization procedure $\mathtt{A}_2$ to seek a global solution of the GOP (1.1), starting from the solution found by $\mathtt{A}_1$. Alternatively, exact penalty functions [32] can be used. An exact penalty function $\rho : \mathbb{R}^n \rightarrow \mathbb{R}$ is such that, for certain choices of the parameter δ_ρ in (1.22), solving Problem (1.23) yields the same solutions to those of the GOP (1.1)[4]. Hence, the global optimization algorithm is run only once. See [33] for the application of exact penalty functions in the context of GO.

[4]Note that the quadratic penalty function in (1.21) is not an exact penalty function [100].

Chapter 2. Black-box optimization

This Chapter is devoted to reviewing the Black-Box Optimization (BBO) framework, which is a specific instance of global optimization that makes the following assumptions on the GOP (1.1).

Assumption 2.1 (Assumptions for black-box optimization)**.** *Consider the global optimization problem in* (1.1) *with* Ω *defined as in* (1.5)*. In the context of black-box optimization, the cost function and (possibly) some of the constraints functions of the GOP* (1.1) *are assumed to be unknown (in a sense that no analytical formulation is available) and expensive to measure. The term "expensive" refers to the fact that a non-negligible amount of resources (such as time) needs to be spent in order to obtain the values of* $f(\cdot)$ *and (possibly) of some* $g^{(j)}(\cdot)$*'s at a given sample* $\boldsymbol{x}_i \in \mathbb{R}^n$*. Typically, in the BBO framework, computer simulations or real-world experiments are required to measure the quantities of interest.*

For the remainder of this book, we will say that a function is a black-box if it is both unknown and expensive to evaluate.

Note that the just seen Assumption is consistent with the definition of sample evaluation in Chapter 1 (Definition 1.1). Black-box cost functions and constraints functions are present in many engineering design problems [2, 135]. Simulators are common tools for describing arbitrarily complex systems and products and are built, for example, from finite-element physics-based models. Running a simulation can take from a few minutes up to several days. At the same time, simulators might depend on some parameters that need to be optimized. This is where black-box optimization procedures come in handy.

Black-box optimization procedures still aim to solve the GOP (1.1); thus, the algorithms described in Chapter 1 could potentially be used as solvers for BBO problems. However, in practice, *the number of sample evaluations required by general-purpose GO algorithms* (such as `DIRECT` [67]) *to find a "decent" solution is (often) excessively high and thus prohibitive under Assumption 2.1. Instead, in the BBO framework, we would rather find a decision vector with a lower degree of optimality accuracy (as in Definition 1.3) but using very few sample evaluations.* Additionally, the curse of dimensionality (Section 1.2.2) is even more relevant for black-box optimization. Recall that the number of sample evaluations required to achieve an ϵ-optimal solution in the worst case is exponential in n. For this reason, BBO optimization methods typically only consider problems with few decision variables, in general up to $n = 20$ [45].

This Chapter is organized as follows. Section 2.1 defines the black-box optimization problem and shows its relationship to the GOP (1.1). Then, Section 2.2 introduces surrogate-based methods (or response surface techniques), which are the most popular algorithms for solving black-box optimization problems. After that, the subsequent Sections describe response surface techniques in greater detail. In particular, Section 2.3 covers the data used by surrogate-based methods when solving the black-box optimization problem; Section 2.4 delves into two well-known experimental designs, namely full factorial designs and latin hypercube designs; Section 2.5 describes the most commonly used surrogate models, which are based either on radial basis functions or Gaussian processes; lastly, Section 2.6 covers a plethora of infill sampling criteria for different response surface techniques.

2.1 The black-box optimization problem

The global optimization problem in (1.1) *and the black-box optimization problem are mathematically equivalent; the difference lies in how the cost function and (possibly) some of the constraints functions are measured (see Assumption 2.1).* Throughout this book, we explicitly distinguish between the constraints functions whose analytical formulations are known from those which are black-boxes. Hence, we re-write the GOP (1.1) as:

$$\begin{aligned} \mathcal{X}^* &= \arg\min_{\boldsymbol{x}} f(\boldsymbol{x}) \\ \text{s.t.} &\quad \boldsymbol{x} \in \Omega \cap \Xi, \end{aligned} \tag{2.1}$$

where Ω, defined either as in (1.3) or in (1.5)[1], is a set of completely known constraints. Instead, Ξ can either be $\Xi = \mathbb{R}^n$, if no black-box constraints are present, or

$$\Xi = \left\{\boldsymbol{x} : \boldsymbol{g}_\Xi(\boldsymbol{x}) \leq \boldsymbol{0}_{q_\Xi}\right\} \tag{2.2}$$

otherwise. In (2.2), $\boldsymbol{g}_\Xi(\boldsymbol{x})$ is a vector-valued function such that $\boldsymbol{g}_\Xi : \mathbb{R}^n \to \mathbb{R}^{q_\Xi}, q_\Xi \in \mathbb{N}$, and $\boldsymbol{g}_\Xi(\boldsymbol{x}) = \begin{bmatrix} g_\Xi^{(1)}(\boldsymbol{x}) & \dots & g_\Xi^{(q_\Xi)}(\boldsymbol{x}) \end{bmatrix}^\top$. The set of global minimizers of the GOP (2.1) is:

$$\mathcal{X}^* = \left\{\boldsymbol{x}_i^* : \boldsymbol{x}_i^* \in \Omega \cap \Xi, f\left(\boldsymbol{x}_i^*\right) = f^* = \min_{\boldsymbol{x} \in \Omega \cap \Xi} f(\boldsymbol{x}), i = 1, \dots, N^*\right\}, \tag{2.3}$$

which, differently from $\mathcal{X}^*$ in (1.2), also takes into account the constraint set Ξ in (2.2). In any case, we make the following Assumption to potentially avoid a degenerate case for which $\mathcal{X}^* = \emptyset$ in Problem (2.1).

Assumption 2.2. *Throughout this book, we assume that* $\Omega \cap \Xi \neq \emptyset$ *for the GOP* (2.1).

Depending on structure of Ξ, we can distinguish two black-box optimization frameworks.

[1]Typically, no equality constraints are considered in black-box optimization.

Definition 2.1: Unconstrained and constrained BBO. *Regardless of the structure of* Ω *for the GOP* (2.1)*, we say that a method belongs to the* *unconstrained black-box optimization framework* *if it assumes that* $\Xi = \mathbb{R}^n$*. Vice-versa, whenever* Ξ *is defined as in* (2.2)*, we speak of* *constrained black-box optimization.*

We continue our dissertation on black-box constraints by defining the Ξ-feasibility function as follows.

Definition 2.2: Ξ***-feasibility function.*** *Let* Ξ *be the set that describes the black-box constraints of the GOP* (2.1)*. We define the* Ξ*-feasibility function* $u_\Xi : \mathbb{R}^n \to \{0, 1\}$ *as follows:*

$$u_\Xi(\boldsymbol{x}) = \begin{cases} 1 & \text{if } \boldsymbol{x} \in \Xi \\ 0 & \text{if } \boldsymbol{x} \notin \Xi \end{cases}. \tag{2.4}$$

$u_\Xi(\boldsymbol{x})$ in (2.4) is particularly relevant for those black-box optimization problems for which we can only evaluate whether a sample $\boldsymbol{x}_i \in \mathbb{R}^n$ is feasible with respect to Ξ or not, i.e. when the measures of $\boldsymbol{g}_\Xi(\boldsymbol{x}_i)$ are not available and we can only assess $u_\Xi(\boldsymbol{x}_i)$.

Notation and conventions. We say that a sample $\boldsymbol{x}_i \in \mathbb{R}^n$ is Ξ-feasible if $u_\Xi(\boldsymbol{x}_i) = 1$ and Ξ-infeasible if $u_\Xi(\boldsymbol{x}_i) = 0$. Moreover, $\boldsymbol{x}_i$ is feasible (with respect to the GOP (2.1)) if $\boldsymbol{x}_i \in \Omega \cap \Xi$ and infeasible otherwise.

In practice, when black-box constraints are present, the goal of finding the global minimizers of the GOP (2.1) becomes quite far-fetched, as pointed out by the following Remark.

Remark 2.1 (On finding the global minima of a constrained black-box optimization problem [111])**.** *The ideal goal of any constrained black-box optimization procedure is to find a suitably accurate global minimizer of the GOP* (2.1) *(i.e. with a realistic optimality accuracy* ϵ *as in Definition 1.3), using relatively few sample evaluations. However, even for low-dimensional unconstrained black-box optimization problems, a large number of sample evaluations is required to guarantee that the obtained solution is even approximately optimal. For higher-dimensional black-box constrained problems, finding a suitably accurate global minimizer of the GOP* (2.1) *within a reasonable amount of sample evaluations becomes impossible. Hence, a more realistic goal followed by most constrained BBO algorithms is: seek a good local minima of the GOP* (2.1)*, possibly starting from a point (or a set of points) that is (are)* Ξ*-feasible.*

We add that, consistently with Remark 2.1, no convergence proofs are available for most black-box optimization algorithms in the constrained BBO framework, since it is not a major concern. In practice, the easiest way for a BBO procedure to ensure its global convergence to the global minimizer(s) of the GOP (2.1) (even in the presence of black-box constraints) is to generate a sequence of iterates $\langle \boldsymbol{x}_i \rangle_{i \geq 1}$ that is dense in Ω. Consequently, this makes $\langle \boldsymbol{x}_i \rangle_{i \geq 1}$ also dense in $\Omega \cap \Xi$. Finally, if $f(\boldsymbol{x})$ is continuous and both Ω and Ξ are compact, such an algorithm would converge to the global minima of the GOP (2.1) (cf. Theorem 1.1 and Theorem 1.2). *There is a clear downside to this approach: if the size of Ξ is small in comparison to Ω, then an algorithm that produces a sequence of iterates that is dense in Ω is bound to spend most of its sample evaluations on points that are not Ξ-feasible (at least, in the long run).* In practice, when dealing with black-box constrained problems, we must limit the evaluation of Ξ-infeasible samples as much as possible. For example, a point $\boldsymbol{x}_i \notin \Xi$ could be associated to a controller calibration that results in an unstable closed-loop system or in extremely unsatisfactory performances, which must be avoided at all costs.

Constraints for black-box optimization can be classified according to the QRAK taxonomy, which we now review.

Definition 2.3: QRAK taxonomy [6, 35]. *We classify a generic constraint,*

$$\tilde{g}(\boldsymbol{x}) \leq 0, \tag{2.5}$$

$\tilde{g} : \mathbb{R}^n \to \mathcal{A}, \mathcal{A} \subseteq \mathbb{R}$, *for the GOP* (2.1) *as follows:*

1. *Quantifiable or binary:*
   - *Quantifiable: it is possible to measure the degree of feasibility and/or violation of the constraint, for instance $\tilde{g} : \mathbb{R}^n \to \mathbb{R}$;*
   - *Binary: given a sample $\boldsymbol{x}_i \in \mathbb{R}^n$, we only know if the constraint in* (2.5) *is violated or not. A binary constraint function is defined as $\tilde{g} : \mathbb{R}^n \to \{0, 1\}$, resulting either in $\tilde{g}(\boldsymbol{x}_i) = 0$ (feasible) or $\tilde{g}(\boldsymbol{x}_i) = 1$ (infeasible).*
2. *Relaxable or unrelaxable:*
   - *Relaxable: a relaxable constraint is one that does not need to be satisfied in order to obtain meaningful outputs from the simulations/experiments that are performed to measure the black-box cost function $f(\boldsymbol{x})$;*
   - *Unrelaxable: the opposite of before. For example, suppose that we want to optimize a certain positive physical quantity; it would not make sense to simulate its corresponding physics-based model using a negative value for it, since we would obtain a measure of $f(\boldsymbol{x})$ that is meaningless.*

3. *A-priori vs black-box:*

    - *A-priori: the analytical expression of function $\tilde{g}(\boldsymbol{x})$ is available;*
    - *Black-box: $\tilde{g}(\boldsymbol{x})$ is unknown but its values can be obtained from expensive computer simulations or real-world experiments (cf. Assumption 2.1).*

4. *Known vs hidden:*

    - *Known: the constraint $\tilde{g}(\boldsymbol{x}) \leq 0$ is explicitly given to the optimization solver (in practice, it could be a black-box, but we know of its existence beforehand);*
    - *Hidden: the constraint $\tilde{g}(\boldsymbol{x}) \leq 0$ is unknown to the solver (not taken into consideration). For example, if $f(\boldsymbol{x})$ is measured from a computer simulation, a hidden constraint could be associated to simulation crashes.*

5. *Coupled vs decoupled [49]:*

    - *Coupled: the values of $f(\boldsymbol{x})$ and $\tilde{g}(\boldsymbol{x})$ are obtained from the same simulation/experiment;*
    - *Decoupled: $f(\boldsymbol{x})$ and $\tilde{g}(\boldsymbol{x})$ can be measured separately.*

Going back to the GOP (2.1), keeping in mind Definition 2.3, we assume that:

- The constraints defined by the set Ω are unrelaxable, a-priori, known and decoupled. No additional assumption is made regarding whether they are quantifiable or binary;
- Vice-versa, the constraints defined by the set Ξ are relaxable, black-box and known. In practice, we do not strictly assume that the constraints in (2.2) are coupled or decoupled. However, we suppose that a sample evaluation returns all the information of interest on both the cost function and the constraints functions. Finally, some BBO methods assume that it is possible to measure $\boldsymbol{g}_{\Xi}(\cdot)$ (quantifiable), others suppose that we can only asses whether a sample is Ξ-feasible or not (binary).

We conclude this Section with an Example of a black-box optimization problem in the context of control systems. We take advantage of the next Example to introduce some terminology and concepts that will also be relevant for Chapter 8, where we will apply the proposed BBO methods to tune the controllers of a hydraulic forming press.

Example 2.1: Black-box optimization for control systems

The goal of any control system is to correct or limit the deviation of a controlled variable (or *output*), produced by a system under control, from a reference variable (called the *setpoint*). To do so, a *controller* (or regulator) commands suitable corrective actions (namely, the *control actions*) to the system in order to drive the output towards the setpoint [102]. For example, we could be interested in bringing and maintaining the temperature (output) of an oven (system under control) at a certain fixed level (setpoint). To do so, the controller generates the signals (control actions) that are fed to the heaters (*actuators*) to drive the oven temperature towards the desired value.

Different control systems might require different controllers. That depends quite a lot on the performance requirements (which are referred to as *control specifications*) that the regulators must satisfy. For example, if we are controlling a cooking oven, we would like its temperature to reach the desired value in the least amount of time as possible (low rise time) but, simultaneously, avoid temperature peaks that could ruin the food. Additionally, we would prefer to have moderate control actions to limit electricity consumption.

Most regulators depend on some parameters. For example, the widely known Proportional-Integral-Derivative (PID) controllers [4] have three parameters which are, respectively, the proportional, integral and derivative gains. These parameters affect the setpoint tracking performances, hence they must be carefully tuned to satisfy the control specifications. *This is where black-box optimization procedures come in. In control systems applications, the parameters of the controller constitute the decision vector while the cost function and the constraints functions are related to the control specifications.*

Let us consider the output, setpoint and control action signals denoted, respectively, as:

$$output(t;\boldsymbol{x}), \quad SP(t), \quad control\_action(t;\boldsymbol{x}). \tag{2.6}$$

Consistently with the control systems literature, we use the variable t to denote the time. In particular, $SP(t)$ is the value of the setpoint at time $t \in \mathbb{R}$ (continuous time signal). If the signals in (2.6) are sampled at a certain sampling time $T_s \in \mathbb{R}_{>0}$, then $SP(t)$ is the value of the setpoint at time $t \cdot T_s$, where $t \in \mathbb{Z}$ (discrete time signal). Notice that the output and the control action signals in (2.6) depend on the tuning, $\boldsymbol{x}$, of the controller.

A common and easy way to assess the quality of a (closed-loop) control system is to perform a *step test*. In this case, the setpoint is defined as:

$$SP(t) = \begin{cases} 0 & \text{for } t < 0 \\ A & \text{for } t \geq 0 \end{cases}, \tag{2.7}$$

where $A \in \mathbb{R}$, and the resulting $output(t; \boldsymbol{x})$ is called the *step response* of the system. If $SP(t)$ is defined as in (2.7), there exist several indicators that describe the quality of the tracking performances of the controller [36, 102]:

1. The *rise time* $t_{rise}(\boldsymbol{x})$, which is the time required by the output to rise from $0.1 \cdot A$ (10% of the setpoint) to $0.9 \cdot A$ (90% of the setpoint);

2. The *settling time* $t_{settle}(\boldsymbol{x})$, which is the time required by the output to reach and stay within a tolerance of 5% from the setpoint;

3. The *maximum overshoot* $max\_os(\boldsymbol{x})$, i.e. the difference between the maximum peak of the output signal and the setpoint. Typically, it is expressed in percentage with respect to A.

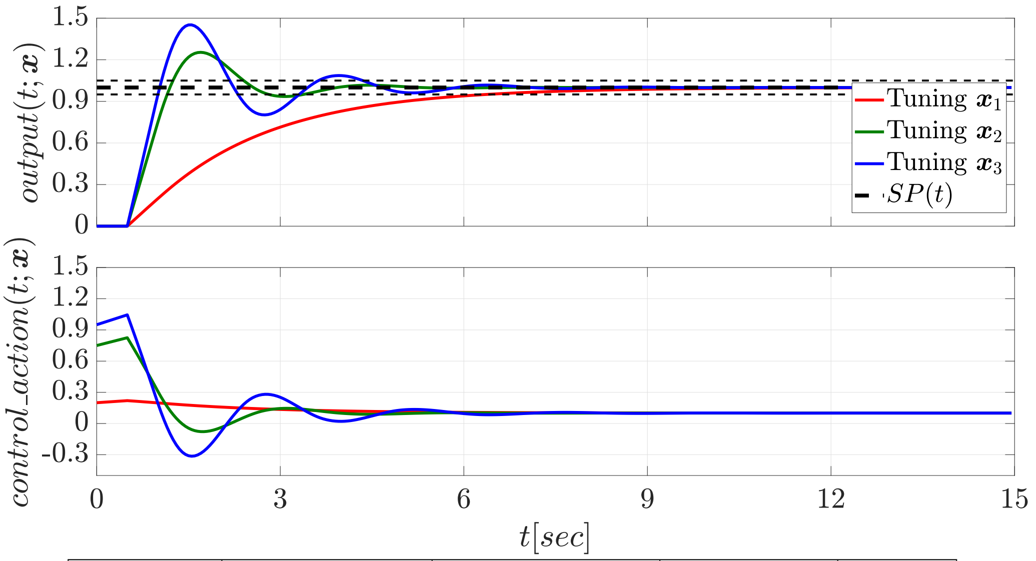
Additionally, one way to measure the "aggressiveness" of the control action is the *Total Square Variation (TSV)* index [36]. Consider $output(t; \boldsymbol{x})$ to be sampled at a sampling time T_s and that a total of $T \in \mathbb{N}$ samples have been acquired. Then, the TSV indicator is defined as:

$$tsv\,(\boldsymbol{x}) = \frac{1}{T_s} \sum_{t=1}^{T-1} \left[control\_action(t+1; \boldsymbol{x}) - control\_action(t; \boldsymbol{x})\right]^2. \tag{2.8}$$

In general, low $tsv\,(\boldsymbol{x})$'s indicate smoother (and thus more moderate) control actions. Vice-versa, high $tsv\,(\boldsymbol{x})$'s are associated to more aggressive regulators.
Figure 3 depicts the values of the proposed indicators for different tunings of a controller. We can perform black-box optimization as follows: we minimize the rise time or the settling time of the step response (i.e. $f\,(\boldsymbol{x}) = t_{rise}(\boldsymbol{x})$ or $f\,(\boldsymbol{x}) = t_{settle}(\boldsymbol{x})$) while keeping the maximum overshoot below a given threshold $\tilde{o} \in \mathbb{R}_{\geq 0}$ (i.e. $\Xi = \{\boldsymbol{x} : max\_os\,(\boldsymbol{x}) \leq \tilde{o}\}$). Even the $tsv\,(\boldsymbol{x})$ indicator in (2.8) could be used as a black-box constraint in a similar fashion. Furthermore, Ω should contain some bounds on the regulators' parameters (for example, often the gains of PID controllers must be non-negative [4, 102]). In this case, a sample evaluation[a] for $\boldsymbol{x}_i$ amounts to: (i) performing a closed-loop step test with controller's parameters $\boldsymbol{x}_i$, acquiring the signals in (2.6), and (ii)

computing the performance indicators of interest from $output(t; \boldsymbol{x}_i)$ and $control\_action(t; \boldsymbol{x}_i)$. Clearly, if no model of the control system is available, the relationship between $t_{rise}(\boldsymbol{x})$, $t_{settle}(\boldsymbol{x})$, $max\_os(\boldsymbol{x})$, $tsv(\boldsymbol{x})$, and the decision vector $\boldsymbol{x}$ is unknown. That is often the case for complex control systems, for which it can be challenging to derive sufficiently expressive models, making black-box optimization procedures suited for the controller calibration task.

[a]Be careful not to confuse the samples of the signals in (2.6) from the samples of the decision vector (tunings) tested by a black-box optimization procedure.



Tuning $\boldsymbol{x}_i$	$t_{rise}(\boldsymbol{x}_i)$ $[sec]$	$t_{settle}(\boldsymbol{x}_i)$ $[sec]$	$max\_os(\boldsymbol{x}_i)$	$tsv(\boldsymbol{x}_i)$
$\boldsymbol{x}_1$ (red)	4.270	6.347	0.000%	0.004
$\boldsymbol{x}_2$ (green)	0.544	3.377	25.369%	0.875
$\boldsymbol{x}_3$ (blue)	0.422	4.348	45.197%	2.530

Figure 3: Performances achieved by a controller with tunings $\boldsymbol{x}_i, i = 1, 2, 3,$ on the same control system (see Example 2.1). The output signal and the control action of each regulator are shown with a different color. The dashed black lines denote the setpoint and the ±5% band. The table reports the values of the indicators described in Example 2.1.

2.2 The rationale behind surrogate-based methods

Now, we review some algorithms that can be used to solve the GOP (2.1). As pointed out in Section 1.2, surrogate-based methods (or response surface techniques) [65, 149] are the de facto standard algorithms for black-box (and preference-based) optimization. Surrogate-based methods are iterative procedures which typically follow the scheme reported in Algorithm 5. We can distinguish three main phases [149]:

1. The experimental design phase [91, 123, 135], which is devoted to generating a set of $N_{init} \in \mathbb{N}$ initial samples that is "well-spread" within the constraint set Ω (see Section 2.4).

2. The following two phases are executed at each iteration of any surrogate-based procedure:

    a) Build/update surrogate model(s) for the cost function and (possibly) the black-box constraints functions (or the Ξ-feasibility function). A surrogate model is none other than an approximation of $f(\boldsymbol{x})$ or of $\boldsymbol{g}_{\Xi}(\boldsymbol{x})/u_{\Xi}(\boldsymbol{x})$ that is constructed using only the data obtained from the sample evaluations. The most popular ones are based either on radial basis functions [38] or Gaussian processes [153] (see Section 2.5).

    b) Infill sampling phase [65, 149], which is devoted to generating new candidate samples to be evaluated. There exist several different infill sampling criteria but, in general, this step amounts to solving an additional global optimization problem where an acquisition function, $a : \mathbb{R}^n \to \mathbb{R}$, is either minimized or maximized. $a(\boldsymbol{x})$ is a function which trades-off *exploitation* (i.e. using the surrogate models as "proxies" for the GOP (2.1)) and *exploration* of the constraint set Ω. See Section 2.6 for a review of some popular infill sampling criteria.

Given the expensiveness of the cost function and, possibly, of some of the constraints functions (see Assumption 2.1), the usual stopping criterion for surrogate-based methods is a maximum number of sample evaluations, $N_{max} \in \mathbb{N}$, which we refer to as budget. For the remainder of this book, we denote the *best candidate sample* found by a surrogate-based method when $N \in \mathbb{N}$ samples have been tried as $\boldsymbol{x_{best}}(N)$. Similarly, the measure of the cost function at $\boldsymbol{x_{best}}(N)$ is $y_{best}(N)$. Then, intuitively, the solution found by any surrogate-based method when the budget is exhausted is $\boldsymbol{x_{best}}(N_{max})$ (with cost $y_{best}(N_{max})$). As we will see in Chapter 3, response surface technique can also be used to solve preference-based optimization problems, following the same scheme described in Algorithm 5.
Clearly, Algorithm 5 is computationally expensive since, at each iteration, the surrogate models must be updated and, also, an additional global optimization problem must be solved. However, most response surface techniques make the following Assumption.

Assumption 2.3 (Assumption for surrogate-based methods)**.** *The computational overhead caused by Step 6 and Step 7 of Algorithm 5 is negligible when compared to the time required to perform sample evaluations.*

Lastly, note that most response surface techniques have several hyper-parameters that need to be carefully tuned. Some of them are related to the surrogate models while others regulate, for example, the exploration-exploitation trade-off and the penalization of Ξ-infeasible regions of Ω. Hence, some surrogate-based procedures also include a recalibration phase devoted to tuning these hyper-parameters (Step 5 of Algorithm 5).

Algorithm 5: General scheme for surrogate-based methods

Input: (i) A-priori known constraint set Ω of the GOP (2.1); (ii) Initial number of samples $N_{init} \in \mathbb{N}$; (iii) Budget $N_{max} \in \mathbb{N}$, $N_{max} > N_{init}$; (iv) (Possibly) hyper-parameters for each phase of the surrogate-based procedure.
Output: (i) (If available) best cost obtained by the procedure $y_{best}(N_{max})$; (ii) Best sample obtained by the procedure $\boldsymbol{x_{best}}(N_{max})$.

```
1:  (Experimental design) Select a set of $N_{init}$ starting points
2:  Evaluate the initial samples
3:  Set $N = N_{init}$
4:  repeat
5:      (Optional) Recalibrate the hyper-parameters
6:      (Build/update surrogate model(s)) Use the available data to construct surrogate models for the cost
        function and the black-box constraints functions/$\Xi$-feasibility function
7:      (Infill sampling phase) Use the previously computed surrogate model(s) to generate one or more new
        candidate samples
8:      Evaluate the new candidate samples
9:      Update the data (i.e. store the new sample evaluations)
10:     Update the best candidate sample (if needed)
11:     Increase $N$ accordingly to how many points have been generated by the infill sampling phase
12: until $N = N_{max}$
```

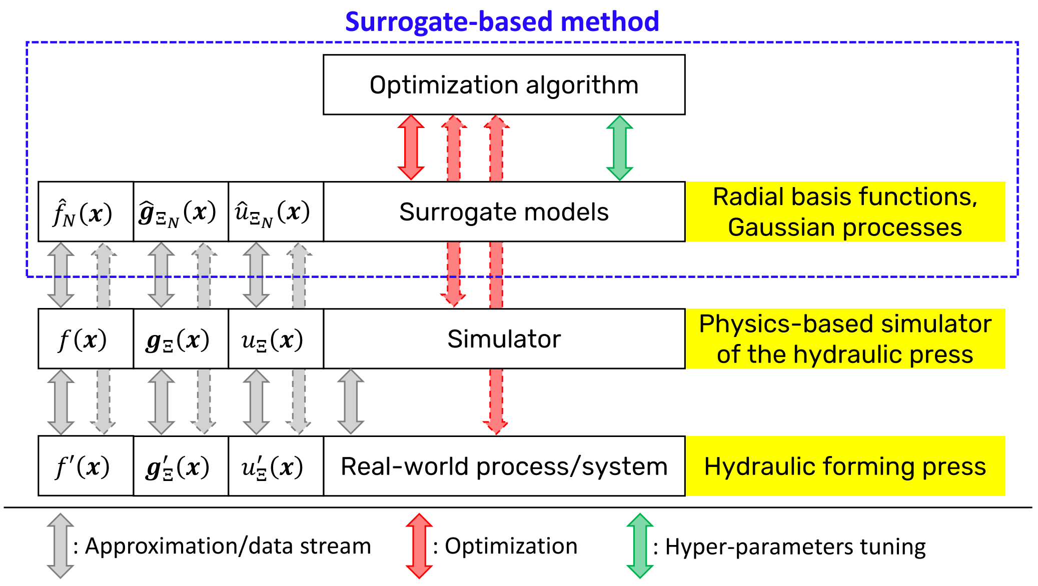

Surrogate-based method
Optimization algorithm
$\hat{f}_N(\boldsymbol{x})$
$\hat{\boldsymbol{g}}_{\Xi_N}(\boldsymbol{x})$
$\hat{u}_{\Xi_N}(\boldsymbol{x})$
Surrogate models
Radial basis functions, Gaussian processes
$f(\boldsymbol{x})$
$\boldsymbol{g}_{\Xi}(\boldsymbol{x})$
$u_{\Xi}(\boldsymbol{x})$
Simulator
Physics-based simulator of the hydraulic press
$f'(\boldsymbol{x})$
$\boldsymbol{g}'_{\Xi}(\boldsymbol{x})$
$u'_{\Xi}(\boldsymbol{x})$
Real-world process/system
Hydraulic forming press
: Approximation/data stream
: Optimization
: Hyper-parameters tuning

Figure 4: General workflow for surrogate-based optimization. A surrogate-based method interacts with either a simulator or a real-world system, acquiring data that is used for the estimation of its surrogate models. The continuous arrows highlight a typical workflow. Instead, the dashed arrows represent a possible (but less likely) scenario of interaction. Lastly, we report some examples for each layer (highlighted in yellow). This Figure has been adapted from [139].

We conclude this Section by giving a *general workflow for surrogate-based optimization* [139], see Figure 4. Following Assumption 2.1, we presume that each sample evaluation amounts to either running a computer simulation or performing a real-world experiment. In the latter case, surrogate-based methods interact directly with the real-world process/system. However, often, a direct interaction with the real system is avoided. Instead, we rely on a simulator that approximates the process to be optimized. Thus, surrogate-based methods acquire information on $f(\boldsymbol{x})$ and, possibly, $\boldsymbol{g}_{\Xi}(\boldsymbol{x})$ and/or

$u_\Xi(\boldsymbol{x})$ by running simulations with certain calibrations of the decision vector. Then, the surrogate models $\hat{f}_N(\boldsymbol{x}), \hat{\boldsymbol{g}}_{\Xi_N}(\boldsymbol{x})$ and $\hat{u}_{\Xi_N}(\boldsymbol{x})$ are estimated from the acquired data (the subscripts indicate that $N \in \mathbb{N}$ sample evaluations have been used for their estimation, as we will see in Section 2.5). The latter are then optimized in some way (through a suitable GO procedure) by the infill sampling criterion of the considered response surface technique. Note that, in theory, we could perform the optimization by interfacing the global optimization procedure directly to either the simulator or the real-world process (dashed red arrows in Figure 4). However, due to the expensiveness of sample evaluations, it is definitely not recommended.

2.3 Data available for black-box optimization

Before moving on to analyze each main phase of Algorithm 5, we highlight the data available to surrogate-based methods after performing several sample evaluations (in the black-box optimization framework). Suppose that $N \in \mathbb{N}$ samples have been evaluated, then we have at our disposal:

1. The locations of the samples:

$$\mathcal{X} = \left\{\boldsymbol{x}_i : i = 1, \dots, N, \boldsymbol{x}_i \in \Omega, \boldsymbol{x}_i \neq \boldsymbol{x}_j, \forall i \neq j\right\}. \tag{2.9}$$

   Note that, in the black-box (and preference-based) optimization framework(s), all samples are generated so that they are feasible with respect to the constraint set Ω of the GOP (2.1). Moreover, in (2.9), we have assumed that all points are different, although it might not necessarily be the case if the experimental design phase and/or the infill sampling phase are not properly defined.

2. The cost function measures:

$$\mathcal{Y} = \Big\{y_i : y_i = f(\boldsymbol{x}_i) + \eta_{f_i}, \boldsymbol{x}_i \in \mathcal{X}, \tag{2.10}$$
$$\eta_{f_i} \overset{i.i.d.}{\sim} \text{some probability distribution}\Big\}.$$

   In general, the measures of $f(\boldsymbol{x})$ can be affected by noise, as highlighted by (2.10).

3. The information on the black-box constraints in Ξ. In practice, we suppose that, at least, we know the feasibility of each sample in $\mathcal{X}$ with respect to Ξ, as highlighted by the following set:

$$\mathcal{U}_\Xi = \{u_i : u_i = u_\Xi(\boldsymbol{x}_i), \boldsymbol{x}_i \in \mathcal{X}\}. \tag{2.11}$$

Most surrogate-based algorithms also assume that it is possible to measure the values of each constraint function $g_{\Xi}^{(j)}(\boldsymbol{x}_i), \forall \boldsymbol{x}_i \in \mathcal{X}, \forall j = 1, \dots, q_{\Xi}$:

$$C_{\Xi} = \Big\{\boldsymbol{c}_i : \boldsymbol{c}_i = \boldsymbol{g}_{\Xi}(\boldsymbol{x}_i) + \boldsymbol{\eta}_{\Xi_i}, \boldsymbol{x}_i \in \mathcal{X}, \quad \boldsymbol{\eta}_{\Xi_i} \overset{i.i.d.}{\sim} \text{some probability distribution}\Big\}. \tag{2.12}$$

Similarly to the cost function, the constraints functions measures in (2.12) might also be affected by noise. In any case, we assume that the noise terms affecting each $g_{\Xi}^{(j)}(\boldsymbol{x})$ are independent. Thus, we can split set C_{Ξ} into q_{Ξ} different sets, one for each black-box constraint:

$$C_{\Xi}^{(j)} = \Big\{c_i^{(j)} : c_i^{(j)} = g_{\Xi}^{(j)}(\boldsymbol{x}_i) + \eta_{g_{\Xi_i}^{(j)}}, \boldsymbol{x}_i \in \mathcal{X}, \quad \eta_{g_{\Xi_i}^{(j)}} \overset{i.i.d.}{\sim} \text{some probability distribution}\Big\}. \tag{2.13}$$

On a side note, if the measures of the black-box constraints functions are assumed to be affected by noise, a more proper definition for the Ξ-feasibility function in (2.4) would be:

$$u_{\Xi}(\boldsymbol{x}_i) = \begin{cases} 1 & \text{if } \boldsymbol{g}_{\Xi}(\boldsymbol{x}_i) + \boldsymbol{\eta}_{\Xi_i} \leq \boldsymbol{0}_{q_{\Xi}} \\ 0 & \text{if } \boldsymbol{g}_{\Xi}(\boldsymbol{x}_i) + \boldsymbol{\eta}_{\Xi_i} > \boldsymbol{0}_{q_{\Xi}} \end{cases}. \tag{2.14}$$

Hence, the elements of set $\mathcal{U}_{\Xi}$ in (2.11) could be mislabeled. Clearly, the entries of $\mathcal{U}_{\Xi}$ in (2.11) can be derived immediately from C_{Ξ} in (2.12), if the latter set is available.

We remark that all the aforementioned sets have the same cardinality, namely $|\mathcal{X}| = |\mathcal{Y}| = |C_{\Xi}| = |\mathcal{U}_{\Xi}| = N$.
Consistently with [54, 111, 115, 116], we make the following Assumption on the measures of the cost function and the black-box constraints functions.

Assumption 2.4 (Noiseless measures)**.** *Unless stated otherwise, we assume that the measures of* $f(\boldsymbol{x})$ *in* $\mathcal{Y}$ (2.10) *and the measures of* $\boldsymbol{g}_{\Xi}(\boldsymbol{x})$ *in* C_{Ξ} (2.12) *are not affected by noise. That is often the case for deterministic computer simulations, although it might not necessarily be true for real-world experiments.*

2.4 Experimental designs

The experimental design phase of Algorithm 5 is used to find an initial set of samples, $\mathcal{X}, |\mathcal{X}| = N_{init}, N_{init} \in \mathbb{N}$, for evaluation. In this context, Ω in (2.1) is referred to as the design space whereas the output of the procedure, $\mathcal{X}$, is called a design. Consistently with Assumption 2.4, in this Section we briefly cover some *experimental design techniques for deterministic experiments*, i.e. such that, given a sample $\tilde{\boldsymbol{x}} \in \Omega$, replicating the evaluation at $\tilde{\boldsymbol{x}}$ yields the same results [123, 124]. That is often the case for computer simulations, for which no repetition of the evaluations is needed. Conversely, real-world experiments are often affected by different sources of random errors (such as human error, systematic error and noise [135]). In this case, the effects of the random errors can be mitigated by replicating experiments [41].

The most important properties for a good (deterministic) experimental design are [149]:

- Space-fill: the design points should be uniformly spread over the entire design space. That is because most surrogate models are more accurate in the vicinity of those samples from which they are built. Hence, a uniform level of model accuracy throughout the design space requires a uniform spread of points [135].

- Non-collapse: two design points should not share any coordinate value if we do not know a-priori which dimensions are important, i.e. $x_i^{(k)} \neq x_j^{(k)}, \forall \boldsymbol{x}_i, \boldsymbol{x}_j \in \mathcal{X}, i \neq j, 1 \leq k \leq n$. *If all the decision variables of the GOP* (2.1) *are assumed to be relevant, then the non-collapse property becomes less significant.*

- Constraints-fill: the samples generated by the experimental design should satisfy all the constraints in Ω. Most procedures assume that the design space is a simple box; an easy (but inefficient) way to achieve the constraint-fill property is to repeat the experimental design until $\boldsymbol{x}_i \in \Omega, \forall \boldsymbol{x}_i \in \mathcal{X}$.

In what follows, we consider Ω to be defined as in (1.3) (i.e. only box constraints are present). The most straightforward way to sample points in a uniform manner is through a full factorial design [135], which basically boils down to grid search. Each decision variable $x^{(j)}$ is discretized so that it can assume $n_g \geq 1, n_g \in \mathbb{N}$, possible values that are equidistant in the interval $\left[l^{(j)}, u^{(j)}\right]$. Differently from Algorithm 1, full factorial designs typically select the values as the midpoints of each subinterval instead of its bounds, i.e. $x^{(j)}$ can assume the values:

$$l^{(j)} + k \cdot \frac{u^{(j)} - l^{(j)}}{n_g + 1}, \quad k = 1, \dots, n_g. \tag{2.15}$$

Clearly, *full factorial designs are space-filling* but are quite restrictive: N_{init} can only be chosen as $N_{init} = \left(n_g\right)^n$, making them impractical for higher-dimensional problems. An example of full factorial design for $N_{init} = 9$ is depicted in Figure 5.

A better alternative to full factorial designs are Latin Hypercube Designs (LHDs) [91], *which are non-collapsing by construction.* The decision variables are still discretized but, this time, they can assume the values[2]:

$$l^{(j)} + k \cdot \frac{u^{(j)} - l^{(j)}}{N_{init} + 1}, \quad k = 1, \dots, N_{init}. \tag{2.16}$$

For the sake of simplicity, denote these values as $\{1, 2, \dots, N_{init}\}$ (which correspond to k in the previous expression). A LHD generates a matrix of samples $X \in \mathbb{R}^{N_{init} \times n}$ such that every column is a random permutation of $\begin{bmatrix} 1 & 2 & \dots & N_{init} \end{bmatrix}^\top$. Furthermore, X is built so that each value for $x^{(j)}, j = 1, \dots, n$, appears at most once in every column and every row. Figure 5 shows three sets of samples generated by a LHD. Note that *latin hypercube designs are not necessarily space-filling*, e.g. that is not the case if the samples in $\mathcal{X}$ are distributed along the main diagonal of the box defined by Ω. Before moving on, we report a numerical example to better grasp how LHDs work.

Example 2.2: Latin hypercube design

Consider $\Omega = \{\boldsymbol{x} : \boldsymbol{l} \leq \boldsymbol{x} \leq \boldsymbol{u}\}$ with $\boldsymbol{l} = \boldsymbol{0}_2$ and $\boldsymbol{u} = \begin{bmatrix} 1 & 2 \end{bmatrix}^\top$. Suppose that we want to generate $N_{init} = 4$ samples through a latin hypercube design. The values that each decision variable can assume are $\frac{k}{5}$ for $x^{(1)}$ and $\frac{2 \cdot k}{5}$ for $x^{(2)}$, $k = 1, \dots, N_{init}$, i.e.:

$$x^{(1)} \in \{0.2, 0.4, 0.6, 0.8\} \text{ and } x^{(2)} \in \{0.4, 0.8, 1.2, 1.6\}.$$

Re-define the set of values for the two decision variables as simply $\{1, 2, 3, 4\}$. Then, one possible matrix of samples, $X \in \mathbb{R}^{4 \times 2}$, returned by a LHD is obtained by selecting the following random permutations:

$$\begin{bmatrix} 1 & 2 \\ 3 & 1 \\ 2 & 4 \\ 4 & 3 \end{bmatrix} \text{ which amounts to } X = \begin{bmatrix} 0.2 & 0.8 \\ 0.6 & 0.4 \\ 0.4 & 1.6 \\ 0.8 & 1.2 \end{bmatrix}.$$

As previously stated, latin hypercube designs are non-collapsing by construction but not necessarily space-filling. The latter property results from a proper selection of the n permutations of $\begin{bmatrix} 1 & 2 & \dots & N_{init} \end{bmatrix}^\top$ among the $N_{init}!$ choices; in total, there are $(N_{init}!)^n$ different ways of constructing the matrix X, making the LHD space quite vast. In practice, *we seek those designs that minimize some cost function that is a measure of "space-fillingness"*. In [95], the authors propose to evaluate

[2]Some LHDs actually consider (without loss of generality) Ω to be the unit hypercube. Then, instead of selecting the values for each $x^{(j)}$ as in (2.16), N_{init} intervals are considered: $\left[0, \frac{1}{N_{init}}\right], \left[\frac{1}{N_{init}}, \frac{2}{N_{init}}\right], \dots, \left[\frac{N_{init}-1}{N_{init}}, 1\right]$. After that, the possible values for $x^{(j)}$ are extracted randomly from each interval. That is the case for MATLAB's implementation of the LHD procedure, which has been used to generate Figure 5.

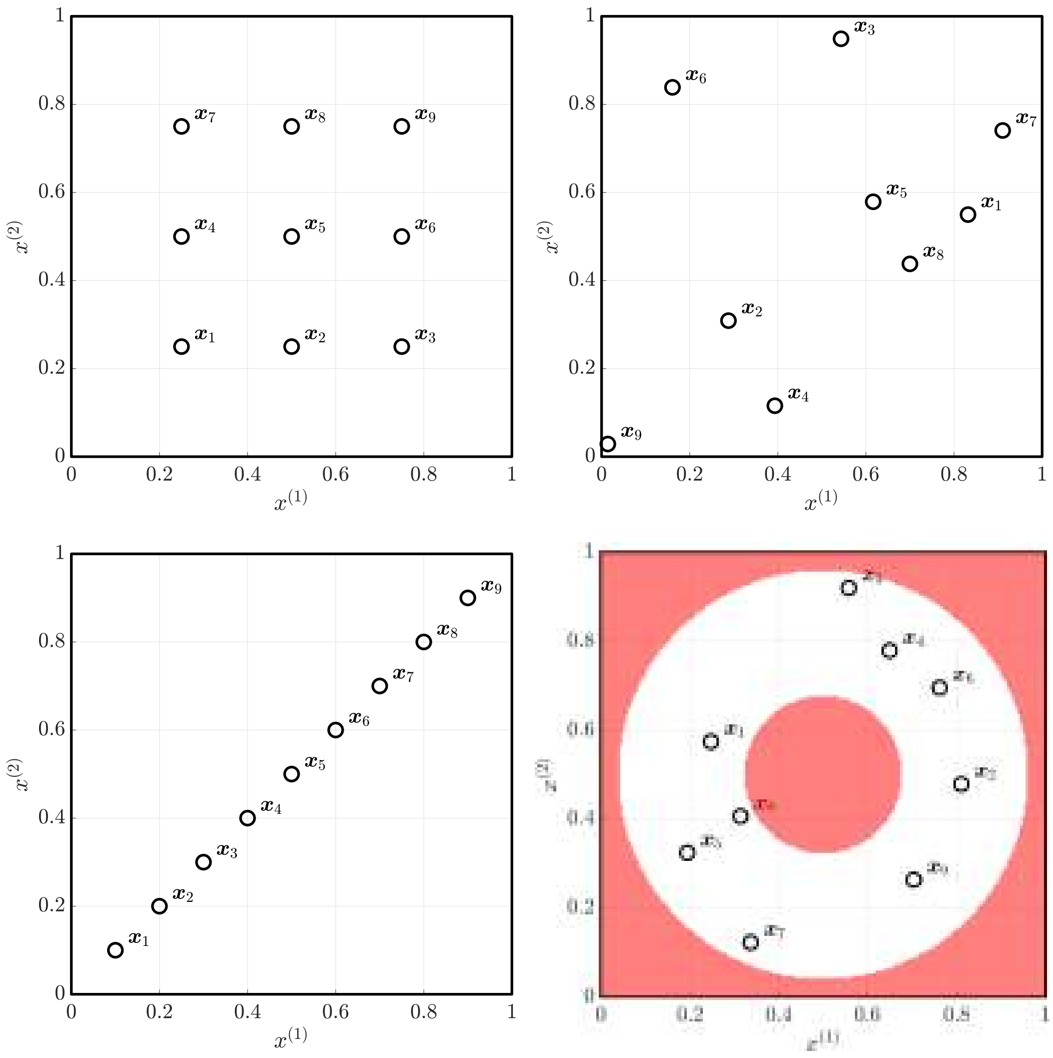


Figure 5: Examples of experimental designs for $n = 2, \boldsymbol{l} = \mathbf{0}_2, \boldsymbol{u} = \mathbf{1}_2$ and $N_{init} = 9$. From left to right, top to bottom: full factorial design, a "good" latin hypercube design (space-filling), a "bad" LHD (not space-filling) and, lastly, a constraints-filling latin hypercube design. The samples produced by the experimental designs are denoted using black circles whereas the infeasible region of the design space Ω is highlighted in red.

the quality of the set of samples $\mathcal{X}$ returned by an experimental design as:

$$J_p(\mathcal{X}) = \left(\sum_{\boldsymbol{x}_i, \boldsymbol{x}_j \in \mathcal{X}, i \neq j} \frac{1}{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^p} \right)^{\frac{1}{p}}, \tag{2.17}$$

where $p \in \mathbb{N}$ is a parameter specified by the user and $\|\cdot\|$ is some norm that defines the distance between the points in $\mathcal{X}$. In the case of latin hypercube designs, $J_p(\mathcal{X})$ in (2.17) is often minimized by means of simulated annealing or evolutionary strategies that are specifically tailored for LHDs, see [95, 135].

So far, we have addressed how to make latin hypercube designs, which are already non-collapsing, also space-filling. For what concerns the constraints-fill property, an easy way to select a constraints-filling LHD is to keep generating sets of samples, $\mathcal{X}$, of increasing size until at least N_{init} points are feasible with respect to Ω. This rationale is described in Algorithm 6. Figure 5 shows an example of constraints-filling latin hypercube design obtained by the proposed scheme. Alternatively, there exist other experimental designs which directly take into account the constraints in Ω [138].

Algorithm 6: Constraints-filling Latin Hypercube Design

Input: (i) Design space Ω, as defined in (1.5); (ii) Number of samples to generate $N_{init} \in \mathbb{N}$.
Output: (i) Design $\mathcal{X}$, $|\mathcal{X}| = N_{init}, \boldsymbol{x}_i \in \Omega, \forall \boldsymbol{x}_i \in \mathcal{X}$.

```
1: Initialize the number of samples to generate as $N_{gen} = N_{init}$
2: Initialize the number of samples that fulfill all the constraints in $\Omega$ as $N_\Omega = 0$
3: while $N_\Omega \neq N_{init}$ do
4:     Generate the set $\mathcal{X}_{gen}$ of $N_{gen}$ samples using a LHD
5:     Extract the samples that satisfy all the constraints, $\mathcal{X} = \mathcal{X}_{gen} \cap \Omega$, and set $N_\Omega = |\mathcal{X}|$
6:     if $N_\Omega > N_{init}$ then
7:         Keep only the first $N_{init}$ samples, i.e. $\mathcal{X} = \{\boldsymbol{x}_i : \boldsymbol{x}_i \in \Omega, i = 1, \dots, N_{init}\}$
8:         Set $N_\Omega = N_{init}$
9:     Increase the number of samples to generate: $N_{gen} = N_{gen} + 1$
```

We conclude this Section by pointing out that there exist several other experimental designs, such as minimax and maximin distance designs [64], as well as statistical designs [123].

2.5 Surrogate models

This Section is devoted to reviewing two of the most popular surrogate models, which are based either on Radial Basis Functions (RBFs) [38] or Gaussian Processes ($\mathcal{GP}$s) [153]. In what follows, we derive the approximations only for the cost function $f(\boldsymbol{x})$ of the GOP (2.1). However, the surrogate models for each black-box constraint function $g_{\Xi}^{(j)}(\boldsymbol{x}), j = 1, \dots, q_\Xi$, can be handled in the same manner, using the measures in $\mathcal{C}_{\Xi}^{(j)}$ (2.13) instead of those in $\mathcal{Y}$ (2.10).

Notation and conventions. Let $h : \mathbb{R}^n \to \mathbb{R}$ be a generic function, we denote a surrogate for $h(\cdot)$ computed from N sample evaluations as $\hat{h}_N : \mathbb{R}^n \to \mathbb{R}$. Furthermore, whenever needed, we explicit the most relevant additional parameters of $\hat{h}_N(\boldsymbol{x})$ using the following notation: $\hat{h}_N(\boldsymbol{x}; \epsilon, \theta)$, where ϵ, θ represent two generic parameters.

The next Remark highlights which surrogate models are most suited for response surface techniques.

Remark 2.2. *Recall that surrogate-based methods aim to solve a global optimization problem. Therefore, most procedures use surrogate models that are able to describe $f(\boldsymbol{x})$ and $g_{\Xi}^{(j)}(\boldsymbol{x}), j = 1, \dots, q_\Xi$, in a global sense. The reasoning behind it is that, if the approximations are "good enough", then solving*

$$\begin{aligned} \hat{\mathcal{X}}^* &= \arg\min_{\boldsymbol{x}} \hat{f}_N(\boldsymbol{x}) \\ s.t. \quad & \boldsymbol{x} \in \Omega \end{aligned}$$

$$\hat{g}^{(j)}_{\Xi_N}(\boldsymbol{x}) \leq 0 \qquad\qquad j = 1, \ldots, q_\Xi$$

leads to solutions that are sufficiently close to the ones of the GOP (2.1).

As a final remark, we stress that the samples in $\mathcal{X}$ (2.9) are assumed to be distinct, which is a mandatory assumption for those surrogate models based on radial basis functions.

2.5.1 Radial basis function interpolation

Before actually defining the surrogate model of interest, we review radial functions and radial basis functions.

Definition 2.4: Radial function [38]. *A function $\phi : \mathbb{R}^n \rightarrow \mathbb{R}$ is called* <u>*radial*</u> *provided that there exists a univariate function $\varphi : \mathbb{R}_{\geq 0} \rightarrow \mathbb{R}$ such that:*

$$\phi(\boldsymbol{x}) = \varphi(r), \quad r = \|\boldsymbol{x}\|, \tag{2.18}$$

where $\|\cdot\|$ is some norm on $\mathbb{R}^n$.

Definition 2.5: Radial basis function [38]. *Given a fixed* <u>*center*</u> *$\boldsymbol{x}_i \in \mathbb{R}^n$, a* <u>*shape parameter*</u> *$\epsilon \in \mathbb{R}_{>0}$ and a radial function $\varphi : \mathbb{R}_{\geq 0} \rightarrow \mathbb{R}$, we define the corresponding* <u>*Radial Basis Function (RBF)*</u> *$\phi_i : \mathbb{R}^n \rightarrow \mathbb{R}$ as:*

$$\phi_i(\boldsymbol{x}; \epsilon) = \varphi(\epsilon \cdot \|\boldsymbol{x} - \boldsymbol{x_i}\|). \tag{2.19}$$

In this book, we will always use the Euclidean norm $\|\boldsymbol{x} - \boldsymbol{x}_i\|_2$ in (2.19). Note that, from Definition 2.5, *radial basis functions are radially (or spherically) symmetric about their centers*, i.e. given any two points $\boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathbb{R}^n$ such that $\|\boldsymbol{x}_1 - \boldsymbol{x}_i\| = \|\boldsymbol{x}_2 - \boldsymbol{x}_i\|$, we have $\phi_i(\boldsymbol{x}_1) = \phi_i(\boldsymbol{x}_2)$. Several RBFs exist; the next Definition reports some of the most commonly used ones.

Definition 2.6: Commonly used radial functions [44]. *Some commonly used radial functions are ($r = \|\boldsymbol{x}\|$):*

- *Inverse quadratic:* $\varphi(r) = \frac{1}{1+r^2}$;
- *Multiquadratic:* $\varphi(r) = \sqrt{1+r^2}$;
- *Linear:* $\varphi(r) = r$;
- *Gaussian:* $\varphi(r) = \exp\left\{-r^2\right\}$;
- *Thin plate spline:* $\varphi(r) = r^2 \cdot \ln(r)$;

- *Inverse multiquadratic:* $\varphi(r) = \frac{1}{\sqrt{1+r^2}}$.

The corresponding radial basis functions, $\phi_i(\boldsymbol{x}; \epsilon)$, can be obtained by setting $r = \epsilon \cdot \|\boldsymbol{x} - \boldsymbol{x_i}\|$.

Figure 6 depicts the aforementioned radial basis functions with different values of the shape parameter. Smaller values of ϵ lead to "flatter" RBFs, whereas increasing the value of the shape parameter makes $\varphi(\cdot)$ "more peaked". Furthermore, in the case of inverse quadratic, Gaussian and inverse multi-quadratic RBFs, a higher ϵ makes $\varphi(\cdot)$ significantly different from zero only in a small neighborhood of the center x_1 (more local behavior).

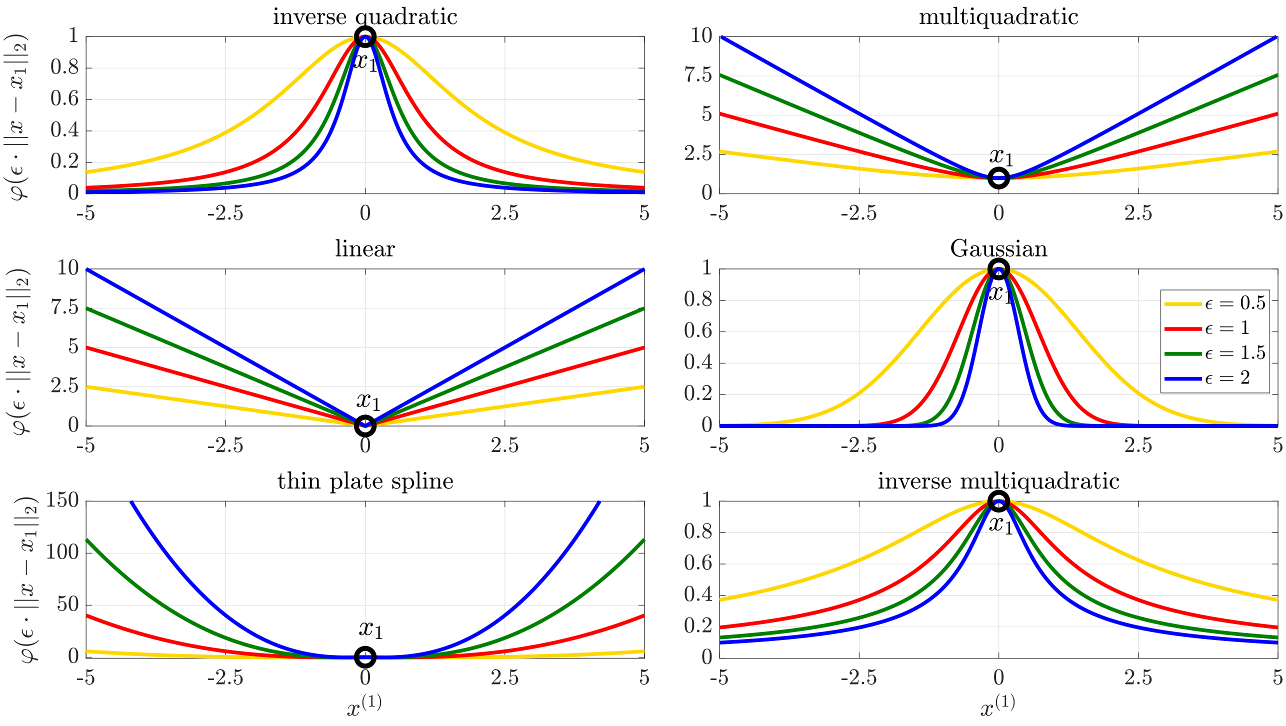


Figure 6: Examples of radial basis functions with center $x_1 = 0$ and for different values of the shape parameter ϵ.

RBFs can be used to approximate the cost function $f(\boldsymbol{x})$ of the GOP (2.1) as follows. Suppose to have at our disposal the data resulting from N sample evaluations (see Section 2.3). We define the surrogate model $\hat{f}_N : \mathbb{R}^n \to \mathbb{R}$ as the <u>radial basis function expansion</u> [38]:

$$\begin{aligned} \hat{f}_N\left(\boldsymbol{x}; \boldsymbol{\beta}_f, \epsilon_f\right) &= \sum_{i=1}^{N} \beta_f^{(i)} \cdot \varphi_f\left(\epsilon_f \cdot \|\boldsymbol{x} - \boldsymbol{x}_i\|_2\right) \\ &= \sum_{i=1}^{N} \beta_f^{(i)} \cdot \phi_{f_i}\left(\boldsymbol{x}; \epsilon_f\right) \\ &= \boldsymbol{\phi}_f\left(\boldsymbol{x}; \epsilon_f\right)^\top \cdot \boldsymbol{\beta}_f, \end{aligned} \tag{2.20}$$

where the subscripts $(\cdot)_f$ indicate that the shape parameter $\epsilon_f \in \mathbb{R}_{>0}$ and the radial function $\varphi_f(\cdot)$ are referred to the surrogate model for the cost function $f(\boldsymbol{x})$. In (2.20), $\boldsymbol{\phi}_f\left(\boldsymbol{x};\epsilon_f\right) \in \mathbb{R}^N$ is the *radial basis function vector*,

$$\begin{aligned}\boldsymbol{\phi}_f\left(\boldsymbol{x};\epsilon_f\right) &= \begin{bmatrix}\varphi_f\left(\epsilon_f \cdot \|\boldsymbol{x}-\boldsymbol{x}_1\|_2\right) & \dots & \varphi_f\left(\epsilon_f \cdot \|\boldsymbol{x}-\boldsymbol{x}_N\|_2\right)\end{bmatrix}^\top \\ &= \begin{bmatrix}\phi_{f_1}\left(\boldsymbol{x};\epsilon_f\right) & \dots & \phi_{f_N}\left(\boldsymbol{x};\epsilon_f\right)\end{bmatrix}^\top,\end{aligned} \tag{2.21}$$

and $\boldsymbol{\beta}_f = \begin{bmatrix}\beta_f^{(1)} & \dots & \beta_f^{(N)}\end{bmatrix}^\top \in \mathbb{R}^N$ is a vector of weights which needs to be computed from the sets $\mathcal{X}$ in (2.9) and $\mathcal{Y}$ in (2.10). The main advantages of the RBF expansion surrogate model in (2.20) are [38]:

1. *It is invariant under all Euclidean transformations* (i.e. translations, rotations and reflections),
2. *It is insensitive to the dimension n of the space $\mathbb{R}^n$ where the samples lie.*

Furthermore, we can state the following results on the differentiability of the RBF expansion surrogate model.

Proposition 2.1: Differentiability of the RBF expansion. *Function $\hat{f}_N\left(\boldsymbol{x};\boldsymbol{\beta}_f,\epsilon_f\right)$ in* (2.20) *is differentiable everywhere with respect to $\boldsymbol{x}$ if and only if the chosen radial basis function $\phi_{f_i}\left(\boldsymbol{x};\epsilon_f\right) = \varphi_f\left(\epsilon_f \cdot \|\boldsymbol{x}-\boldsymbol{x}_i\|_2\right)$ is differentiable everywhere.*

Proof. Clearly, if $\phi_{f_i}\left(\boldsymbol{x};\epsilon_f\right)$ in (2.20) is differentiable everywhere with respect to $\boldsymbol{x}$, then $\hat{f}_N\left(\boldsymbol{x};\boldsymbol{\beta}_f,\epsilon_f\right)$ is a linear combination of differentiable functions, making it differentiable. Vice-versa, if $\phi_{f_i}\left(\boldsymbol{x};\epsilon_f\right)$ is not differentiable everywhere, then the same can be said for the surrogate model in (2.20). □

Lemma 2.1: Gradient of the RBF expansion. *Suppose that $\hat{f}_N\left(\boldsymbol{x};\boldsymbol{\beta}_f,\epsilon_f\right)$ in* (2.20) *is differentiable everywhere (see Proposition 2.1). Then, its gradient is:*

$$\nabla_{\boldsymbol{x}}\hat{f}_N\left(\boldsymbol{x};\boldsymbol{\beta}_f,\epsilon_f\right) = \sum_{i=1}^{N}\beta_f^{(i)} \cdot \nabla_{\boldsymbol{x}}\phi_{f_i}\left(\boldsymbol{x};\epsilon_f\right), \tag{2.22}$$

where $\nabla_{\boldsymbol{x}}\phi_{f_i}\left(\boldsymbol{x};\epsilon_f\right)$ is the gradient of the chosen radial basis function.

We follow up Proposition 2.1 with two Examples where we show that the linear RBF leads to a surrogate model that is not differentiable at each $\boldsymbol{x}_i \in \mathcal{X}$, whereas the inverse quadratic RBF results in a $\hat{f}_N\left(\boldsymbol{x}; \boldsymbol{\beta}_f, \epsilon_f\right)$ in (2.20) that is differentiable everywhere. We also take advantage of the next Examples to show some basic differentiability results on the Euclidean norm and the squared Euclidean norm.

Example 2.3: Linear radial basis function

We want to prove that $\hat{f}_N\left(\boldsymbol{x}; \boldsymbol{\beta}_f, \epsilon_f\right)$ in (2.20) defined using the linear radial basis function, i.e. $\phi_{f_i}\left(\boldsymbol{x}; \epsilon_f\right) = \epsilon_f \cdot \|\boldsymbol{x} - \boldsymbol{x}_i\|_2$, is not differentiable at each $\boldsymbol{x}_i \in \mathcal{X}$. Consider the Euclidean norm $\|\boldsymbol{x} - \boldsymbol{x}_i\|_2$ and define the auxiliary variable $\boldsymbol{z} = \boldsymbol{x} - \boldsymbol{x}_i$. We now prove that $\|\boldsymbol{z}\|_2$ *is differentiable everywhere except at the origin* $\boldsymbol{0}_n$. To do so, we compute the partial derivatives of $\|\boldsymbol{z}\|_2$ at any point $\tilde{\boldsymbol{z}} \in \mathbb{R}^n$:

$$\begin{aligned}
\left.\frac{\partial}{\partial z^{(j)}} \|\boldsymbol{z}\|_2\right|_{\boldsymbol{z}=\tilde{\boldsymbol{z}}} &= \lim_{t\to 0} \frac{\left\|\tilde{\boldsymbol{z}} + t \cdot \boldsymbol{e}_j\right\|_2 - \|\tilde{\boldsymbol{z}}\|_2}{t} \\
&= \lim_{t\to 0} \frac{\left\|\tilde{\boldsymbol{z}} + t \cdot \boldsymbol{e}_j\right\|_2 - \|\tilde{\boldsymbol{z}}\|_2}{t} \cdot \frac{\left\|\tilde{\boldsymbol{z}} + t \cdot \boldsymbol{e}_j\right\|_2 + \|\tilde{\boldsymbol{z}}\|_2}{\left\|\tilde{\boldsymbol{z}} + t \cdot \boldsymbol{e}_j\right\|_2 + \|\tilde{\boldsymbol{z}}\|_2} \\
&= \lim_{t\to 0} \frac{\left\|\tilde{\boldsymbol{z}} + t \cdot \boldsymbol{e}_j\right\|_2^2 - \|\tilde{\boldsymbol{z}}\|_2^2}{t \cdot \left(\left\|\tilde{\boldsymbol{z}} + t \cdot \boldsymbol{e}_j\right\|_2 + \|\tilde{\boldsymbol{z}}\|_2\right)} \\
&= \lim_{t\to 0} \frac{\left(\tilde{\boldsymbol{z}} + t \cdot \boldsymbol{e}_j\right)^\top \cdot \left(\tilde{\boldsymbol{z}} + t \cdot \boldsymbol{e}_j\right) - \tilde{\boldsymbol{z}}^\top \cdot \tilde{\boldsymbol{z}}}{t \cdot \left(\left\|\tilde{\boldsymbol{z}} + t \cdot \boldsymbol{e}_j\right\|_2 + \|\tilde{\boldsymbol{z}}\|_2\right)} \\
&= \lim_{t\to 0} \frac{\cancel{\tilde{\boldsymbol{z}}^\top \cdot \tilde{\boldsymbol{z}}} + t^2 + 2 \cdot t \cdot \tilde{z}^{(j)} - \cancel{\tilde{\boldsymbol{z}}^\top \cdot \tilde{\boldsymbol{z}}}}{t \cdot \left(\left\|\tilde{\boldsymbol{z}} + t \cdot \boldsymbol{e}_j\right\|_2 + \|\tilde{\boldsymbol{z}}\|_2\right)} \\
&= \lim_{t\to 0} \frac{\cancel{t} \cdot \left(t + 2 \cdot \tilde{z}^{(j)}\right)}{\cancel{t} \cdot \left(\left\|\tilde{\boldsymbol{z}} + t \cdot \boldsymbol{e}_j\right\|_2 + \|\tilde{\boldsymbol{z}}\|_2\right)} \\
&= \lim_{t\to 0} \frac{t + 2 \cdot \tilde{z}^{(j)}}{\left\|\tilde{\boldsymbol{z}} + t \cdot \boldsymbol{e}_j\right\|_2 + \|\tilde{\boldsymbol{z}}\|_2}.
\end{aligned}$$

Clearly, if $\tilde{\boldsymbol{z}} \neq \boldsymbol{0}_n$, the partial derivatives at $\tilde{\boldsymbol{z}}$ exist and are equal to:

$$\left.\frac{\partial}{\partial z^{(j)}} \|\boldsymbol{z}\|_2\right|_{\boldsymbol{z}=\tilde{\boldsymbol{z}}} = \frac{\tilde{z}^{(j)}}{\|\tilde{\boldsymbol{z}}\|_2}.$$

Vice-versa, for $\tilde{\boldsymbol{z}} = \boldsymbol{0}_n$, the limit becomes:

$$\begin{aligned}
\lim_{t\to 0} \frac{t + 2 \cdot \tilde{z}^{(j)}}{\left\|\tilde{\boldsymbol{z}} + t \cdot \boldsymbol{e}_j\right\|_2 + \|\tilde{\boldsymbol{z}}\|_2} &= \lim_{t\to 0} \frac{t}{\left\|t \cdot \boldsymbol{e}_j\right\|_2} \\
&= \lim_{t\to 0} \frac{t}{\sqrt{t^2}} \\
&= \lim_{t\to 0} \frac{t}{|t|}.
\end{aligned}$$

We can see that the limit does not exist since:

$$1 = \lim_{t \to 0^+} \frac{t}{|t|} \neq \lim_{t \to 0^-} \frac{t}{|t|} = -1.$$

Thus, we conclude that $\|z\|_2$ is not differentiable at the origin since its partial derivatives do not exist at $z = \mathbf{0}_n$.

To prove that $\|z\|_2$ is differentiable $\forall z \in \mathbb{R}^n \setminus \{\mathbf{0}_n\}$, we still need to verify if all its partial derivatives are continuous on such set. Clearly, that is the case for $\frac{\partial}{\partial z^{(j)}} \|z\|_2 = \frac{z^{(j)}}{\|z\|_2}, \forall j = 1, \ldots, n$.

Now, recall that $z = \boldsymbol{x} - \boldsymbol{x}_i$. Then, *the gradient of* $\|\boldsymbol{x} - \boldsymbol{x}_i\|_2$ *is defined* $\forall \boldsymbol{x} \in \mathbb{R}^n \setminus \{\boldsymbol{x}_i\}$ *and is equal to*:

$$\nabla_{\boldsymbol{x}} \|\boldsymbol{x} - \boldsymbol{x}_i\|_2 = \frac{\boldsymbol{x} - \boldsymbol{x}_i}{\|\boldsymbol{x} - \boldsymbol{x}_i\|_2}. \tag{2.23}$$

Therefore, the linear radial basis function $\phi_{f_i}(\boldsymbol{x}; \epsilon_f) = \epsilon_f \cdot \|\boldsymbol{x} - \boldsymbol{x}_i\|_2$ is not differentiable at $\boldsymbol{x} = \boldsymbol{x}_i$, making $\hat{f}_N\left(\boldsymbol{x}; \boldsymbol{\beta}_f, \epsilon_f\right)$ in (2.20) not differentiable at each $\boldsymbol{x}_i \in \mathcal{X}$.

Example 2.4: Inverse quadratic radial basis function

We want to prove that $\hat{f}_N\left(\boldsymbol{x}; \boldsymbol{\beta}_f, \epsilon_f\right)$ in (2.20) defined using the inverse quadratic radial basis function, i.e. $\phi_{f_i}(\boldsymbol{x}; \epsilon_f) = \left(1 + \epsilon_f^2 \cdot \|\boldsymbol{x} - \boldsymbol{x}_i\|_2^2\right)^{-1}$, is differentiable everywhere. First of all, we demonstrate that *the squared Euclidean norm is differentiable everywhere*. Clearly, $\|\boldsymbol{x} - \boldsymbol{x}_i\|_2^2$ is differentiable $\forall \boldsymbol{x} \in \mathbb{R}^n \setminus \{\boldsymbol{x}_i\}$ since it is the composition of two functions that are differentiable on their respective domains (see Example 2.3). Thus, we only need to check whether or not the squared Euclidean norm is differentiable at $\boldsymbol{x}_i$. We compute its partial derivatives at $\boldsymbol{x}_i$:

$$\begin{aligned}
\left.\frac{\partial}{\partial x^{(j)}} \|\boldsymbol{x} - \boldsymbol{x}_i\|_2^2\right|_{\boldsymbol{x}=\boldsymbol{x}_i} &= \lim_{t \to 0} \frac{\|\boldsymbol{x}_i + t \cdot \boldsymbol{e}_j - \boldsymbol{x}_i\|_2^2 - \|\boldsymbol{x}_i - \boldsymbol{x}_i\|_2^2}{t} \\
&= \lim_{t \to 0} \frac{\|t \cdot \boldsymbol{e}_j\|_2^2}{t} \\
&= \lim_{t \to 0} \frac{t^2}{t} \\
&= 0.
\end{aligned}$$

We have proven that the partial derivates of $\|\boldsymbol{x} - \boldsymbol{x}_i\|_2^2$ exist $\forall \boldsymbol{x} \in \mathbb{R}^n$. Moreover, we can easily compute the analytical expression of each partial derivative by applying the chain rule:

$$\frac{\partial}{\partial x^{(j)}} \|\boldsymbol{x} - \boldsymbol{x}_i\|_2^2 = \left.\frac{d}{dt} t^2\right|_{t=\|\boldsymbol{x}-\boldsymbol{x}_i\|_2} \cdot \frac{\partial}{\partial x^{(j)}} \|\boldsymbol{x} - \boldsymbol{x}_i\|_2$$

$$
\begin{aligned}
&= 2 \cdot t \bigg|_{t=\|\boldsymbol{x}-\boldsymbol{x_i}\|_2} \cdot \frac{x^{(j)} - x_i^{(j)}}{\|\boldsymbol{x} - \boldsymbol{x_i}\|_2} \\
&= 2 \cdot \left(x^{(j)} - x_i^{(j)}\right).
\end{aligned}
$$

Clearly, all $\frac{\partial}{\partial x^{(j)}} \|\boldsymbol{x} - \boldsymbol{x_i}\|_2^2, j = 1, \dots, n$, are continuous everywhere and therefore we can conclude that the squared Euclidean norm is differentiable $\forall \boldsymbol{x} \in \mathbb{R}^n$. Hence, *the gradient of* $\|\boldsymbol{x} - \boldsymbol{x}_i\|_2^2$ *is defined* $\forall \boldsymbol{x} \in \mathbb{R}^n$ *and is equal to*:

$$\nabla_{\boldsymbol{x}} \|\boldsymbol{x} - \boldsymbol{x}_i\|_2^2 = 2 \cdot (\boldsymbol{x} - \boldsymbol{x}_i). \tag{2.24}$$

We now go back to the inverse quadratic radial basis function $\phi_{f_i}\left(\boldsymbol{x}; \epsilon_f\right) = \left(1 + \epsilon_f^2 \cdot \|\boldsymbol{x} - \boldsymbol{x}_i\|_2^2\right)^{-1}$. From the just seen results on $\|\boldsymbol{x} - \boldsymbol{x}_i\|_2^2$, we can conclude that $\phi_{f_i}\left(\boldsymbol{x}; \epsilon_f\right)$ is differentiable everywhere since it is the composition of two functions that are differentiable everywhere on their respective domains:

$$
\begin{aligned}
\phi_{f_i}\left(\boldsymbol{x}; \epsilon_f\right) &= h_1\left(h_2\left(\boldsymbol{x}; \epsilon_f\right)\right), \\
h_1(t) &= \frac{1}{1+t}, & h_1 &: \mathbb{R}_{\geq 0} \to (0, 1], \\
h_2\left(\boldsymbol{x}; \epsilon_f\right) &= \epsilon_f^2 \cdot \|\boldsymbol{x} - \boldsymbol{x_i}\|_2^2, & h_2 &: \mathbb{R}^n \to \mathbb{R}_{\geq 0},
\end{aligned}
$$

and $1 + \epsilon_f^2 \cdot \|\boldsymbol{x} - \boldsymbol{x}_i\|_2^2 > 0, \forall \boldsymbol{x} \in \mathbb{R}^n, \epsilon_f \in \mathbb{R}_{>0}$.
Lastly, we compute the gradient of $\phi_{f_i}\left(\boldsymbol{x}; \epsilon_f\right)$ using the chain rule:

$$
\begin{aligned}
\nabla_{\boldsymbol{x}} \phi_i\left(\boldsymbol{x}; \epsilon_f\right) &= \frac{d}{dt} h_1(t) \bigg|_{t=h_2\left(\boldsymbol{x}; \epsilon_f\right)} \cdot \nabla_{\boldsymbol{x}} h_2\left(\boldsymbol{x}; \epsilon_f\right) \\
&= -\frac{1}{(1+t)^2} \bigg|_{t=h_2(\boldsymbol{x})} \cdot 2 \cdot \epsilon_f^2 \cdot (\boldsymbol{x} - \boldsymbol{x_i}) \\
&= -\frac{2 \cdot \epsilon_f^2}{\left(1 + \epsilon_f^2 \cdot \|\boldsymbol{x} - \boldsymbol{x_i}\|_2^2\right)^2} \cdot (\boldsymbol{x} - \boldsymbol{x_i}).
\end{aligned}
$$

So far, we have defined several radial basis functions and highlighted some key properties for the RBF expansion surrogate model in (2.20), but we have not yet stated how to choose the weight vector $\boldsymbol{\beta}_f$. In practice, *most BBO methods compute* $\boldsymbol{\beta}_f$ *so that* $\hat{f}_N\left(\boldsymbol{x}; \boldsymbol{\beta}_f, \epsilon_f\right)$ *interpolates the data in* $\mathcal{X}$ (2.9) *and* $\mathcal{Y}$ (2.10), namely:

$$\hat{f}_N\left(\boldsymbol{x}_i; \boldsymbol{\beta}_f, \epsilon_f\right) = y_i, \quad \forall i = 1, \dots, N, \boldsymbol{x}_i \in \mathcal{X}, y_i \in \mathcal{Y}.$$

From (2.20), the interpolation conditions amount to:

$$\Phi_f\left(\epsilon_f\right)\cdot\boldsymbol{\beta}_f=\boldsymbol{y}, \tag{2.25}$$

where $\Phi_f\left(\epsilon_f\right)\in\mathbb{R}^{N\times N}$ is a symmetric matrix whose (i,j)-th entry is $\Phi_f^{(i,j)}\left(\epsilon_f\right)=\varphi_f\left(\epsilon_f\cdot\left\|\boldsymbol{x}_i-\boldsymbol{x}_j\right\|_2\right)$, i.e.:

$$\begin{aligned}\Phi_f\left(\epsilon_f\right)&=\begin{bmatrix}\varphi_f\left(\epsilon_f\cdot\|\boldsymbol{x}_1-\boldsymbol{x}_1\|_2\right) & \cdots & \varphi_f\left(\epsilon_f\cdot\|\boldsymbol{x}_1-\boldsymbol{x}_N\|_2\right)\\ \vdots & \ddots & \vdots\\ \varphi_f\left(\epsilon_f\cdot\|\boldsymbol{x}_N-\boldsymbol{x}_1\|_2\right) & \cdots & \varphi_f\left(\epsilon_f\cdot\|\boldsymbol{x}_N-\boldsymbol{x}_N\|_2\right)\end{bmatrix}\\ &=\begin{bmatrix}\phi_{f_1}\left(\boldsymbol{x}_1;\epsilon_f\right) & \cdots & \phi_{f_N}\left(\boldsymbol{x}_1;\epsilon_f\right)\\ \vdots & \ddots & \vdots\\ \phi_{f_1}\left(\boldsymbol{x}_N;\epsilon_f\right) & \cdots & \phi_{f_N}\left(\boldsymbol{x}_N;\epsilon_f\right)\end{bmatrix}\\ &=\begin{bmatrix}\boldsymbol{\phi}_f\left(\boldsymbol{x}_1;\epsilon_f\right)^\top\\ \vdots\\ \boldsymbol{\phi}_f\left(\boldsymbol{x}_N;\epsilon_f\right)^\top\end{bmatrix}.\end{aligned} \tag{2.26}$$

Instead, $\boldsymbol{y}\in\mathbb{R}^N$ in (2.25) is simply a vector which stacks all the cost function measures in $\mathcal{Y}$:

$$\boldsymbol{y}=\begin{bmatrix}y_1 & \dots & y_N\end{bmatrix}^\top. \tag{2.27}$$

Clearly, the linear system in (2.25) has a unique solution if and only if matrix $\Phi_f\left(\epsilon_f\right)$ is non-singular. For some RBFs, matrix $\Phi_f\left(\epsilon_f\right)$ in (2.26) is positive definite (and thus invertible). To analyze the positive definiteness of the interpolation matrix, we introduce the notion of complete monotonicity of functions.

Definition 2.7: Completely monotone function [38, 44]. *A function $\psi:\mathbb{R}_{\geq 0}\to\mathbb{R}$ that is in $C^0\left(\mathbb{R}_{\geq 0}\right)\cap C^\infty\left(\mathbb{R}_{>0}\right)$[a] is said to be completely monotone if and only if*

$$(-1)^k\cdot\frac{d^k}{dr^k}\psi(r)\geq 0,\quad\forall r>0 \text{ and } k=0,1,\dots \tag{2.28}$$

[a]See Definition A.26.

The following Theorem links completely monotone functions to the solution of the linear system in (2.25).

Theorem 2.1: Positive definiteness of $\Phi_f\left(\epsilon_f\right)$ [44]

Let $\psi_f:\mathbb{R}_{\geq 0}\to\mathbb{R}$ be a completely monotone function that is not constant. Then, matrix $\Phi_f\left(\epsilon_f\right)$ in (2.26) based on the radial function $\varphi_f(r)=\psi_f\left(r^2\right)$ is positive definite.

In the next Example we prove that the Gaussian radial basis function gives rise to a positive definite $\Phi_f\left(\epsilon_f\right)$ in (2.26).

Example 2.5: Positive definiteness of the interpolation matrix associated to Gaussian radial basis functions

Consider the Gaussian radial basis function with unitary shape parameter, namely $\varphi_f(r) = \exp\left\{-r^2\right\}$. Following Theorem 2.1, we define $\psi_f : \mathbb{R}_{\geq 0} \to \mathbb{R}$ as:

$$\psi_f(r) = \exp\{-r\}.$$

Clearly, $\varphi_f(r) = \psi_f\left(r^2\right)$. It is easy to prove that the exponential function is infinitely differentiable on $\mathbb{R}$ and:

$$\frac{d^k}{dr^k}\psi_f(r) = (-1)^k \cdot \exp\{-r\}, \quad k = 0, 1, \dots$$

Thus, the conditions in (2.28) are satisfied and $\psi_f(r)$ is a completely monotone function. Finally, by Theorem 2.1, the matrix $\Phi_f\left(\epsilon_f\right)$ in (2.26) originated from the Gaussian radial basis function is positive definite.

There are alternative ways to check whether or not matrix $\Phi_f\left(\epsilon_f\right)$ in (2.26) is positive definite that rely on the concept of positive definite functions, see [23, 38, 44] for more details. Here, we simply highlight which of the radial functions in Definition 2.6 give rise to a non-singular $\Phi_f\left(\epsilon_f\right)$ in (2.26).

Proposition 2.2. *The following radial functions result in a non-singular* $\Phi_f\left(\epsilon_f\right)$ *in* (2.26) *[23, 38, 44]:*

- *Inverse quadratic,*
- *Gaussian (see Example 2.5),*
- *Inverse multiquadratic.*

As a final remark note that, even though $\Phi_f\left(\epsilon_f\right)$ in (2.26) is non-singular for some RBFs, the shape parameter ϵ_f as well as the number and the distribution of the samples in $\mathcal{X}$ (2.9) affect its condition number, see [38, 119] for more details. To compensate for this, the shape parameter ϵ_f is typically chosen through cross-validation [24, 119]. Alternatively, *if exact interpolation is not a concern, then the linear system in* (2.25) *can be solved by taking a low-rank approximation of* $\Phi_f\left(\epsilon_f\right)$, as proposed in [10]. By proceeding this way, any RBF in Definition 2.6 can be used and also the problems related to the condition number of $\Phi_f\left(\epsilon_f\right)$ are accounted for. We will review the low-rank approach for solving the linear system in (2.25) more in detail in Section 4.1.1.

Theorem 2.1 limits the choice of the radial basis functions for $\hat{f}_N\left(\boldsymbol{x}; \boldsymbol{\beta}_f, \epsilon_f\right)$ in (2.20) to only those which lead to a non-singular $\Phi_f\left(\epsilon_f\right)$ in (2.26). An alternative surrogate model which can handle any of the RBFs in Definition 2.6 is [54, 126, 152]:

$$\hat{f}_N\left(\boldsymbol{x}; \boldsymbol{\alpha}_f, \boldsymbol{\beta}_f, \epsilon_f\right) = \boldsymbol{\phi}_f\left(\boldsymbol{x}; \epsilon_f\right)^\top \cdot \boldsymbol{\beta}_f + \pi_f\left(\boldsymbol{x}; \boldsymbol{\alpha}_f\right), \tag{2.29}$$

where $\pi_f\left(\boldsymbol{x}; \boldsymbol{\alpha}_f\right) \in \Pi_{d_\pi}$. Π_{d_π} is the space of polynomials in n variables and of degree at most $d_\pi \in \mathbb{N} \cup \{-1, 0\}$. In particular, if $d_\pi = -1$, then $\Pi_{-1} = \{0\}$ and thus the model in (2.29) is equal to the one in (2.20). Note that $\pi_f\left(\boldsymbol{x}; \boldsymbol{\alpha}_f\right)$ can be expressed as a linear combination of functions $\left[x^{(1)}\right]^{d_1}, \dots, \left[x^{(n)}\right]^{d_n}$ with $d_1 + \ldots + d_n \leq d_\pi$. More formally, consider a basis $\pi_{f_1}\left(\boldsymbol{x}\right), \dots, \pi_{f_{N_\pi}}\left(\boldsymbol{x}\right)$ of Π_{d_π}, with $N_\pi = \begin{pmatrix} d_\pi + n \\ n \end{pmatrix}$. We can write $\pi_f\left(\boldsymbol{x}; \boldsymbol{\alpha}_f\right)$ in (2.29) as:

$$\pi_f\left(\boldsymbol{x}; \boldsymbol{\alpha}_f\right) = \sum_{j=1}^{N_\pi} \alpha_f^{(j)} \cdot \pi_{f_j}\left(\boldsymbol{x}\right), \tag{2.30}$$

where $\boldsymbol{\alpha}_f = \begin{bmatrix} \alpha_f^{(1)} & \dots & \alpha_f^{(N_\pi)} \end{bmatrix}^\top \in \mathbb{R}^{N_\pi}$ is a vector of weights associated to the latter polynomial. In the case of model (2.29), the interpolation conditions amount to:

$$\begin{bmatrix} \Phi_f\left(\epsilon_f\right) & P_f \\ P_f^\top & 0_{N_\pi \times N_\pi} \end{bmatrix} \cdot \begin{bmatrix} \boldsymbol{\beta}_f \\ \boldsymbol{\alpha}_f \end{bmatrix} = \begin{bmatrix} \boldsymbol{y} \\ \boldsymbol{0}_{N_\pi} \end{bmatrix}, \tag{2.31}$$

where $P_f \in \mathbb{R}^{N \times N_\pi}$ is defined as:

$$\begin{aligned} P_f &= \begin{bmatrix} \pi_{f_1}\left(\boldsymbol{x}_1\right) & \cdots & \pi_{f_{N_\pi}}\left(\boldsymbol{x}_1\right) \\ \vdots & \ddots & \vdots \\ \pi_{f_1}\left(\boldsymbol{x}_N\right) & \cdots & \pi_{f_{N_\pi}}\left(\boldsymbol{x}_N\right) \end{bmatrix} \\ &= \begin{bmatrix} \boldsymbol{\pi}_f\left(\boldsymbol{x}_1\right)^\top \\ \vdots \\ \boldsymbol{\pi}_f\left(\boldsymbol{x}_N\right)^\top \end{bmatrix} \end{aligned}$$

and $\boldsymbol{\pi}_f\left(\boldsymbol{x}_i\right) = \begin{bmatrix} \pi_{f_1}\left(\boldsymbol{x}_i\right) & \dots & \pi_{f_{N_\pi}}\left(\boldsymbol{x}_i\right) \end{bmatrix}^\top \in \mathbb{R}^{N_\pi}$. In [54, 107], the authors derive which values of d_π guarantee that the linear system of equations in (2.31) admits a unique solution for each RBF in Definition 2.6, as highlighted by the following Proposition.

Proposition 2.3. *Consider the RBFs in Definition 2.6. Then, selecting* $d_\pi \geq d_{\pi_{min}}$ *for* $\pi_f\left(\boldsymbol{x}; \boldsymbol{\alpha}_f\right)$ *in* (2.29) *as:*

- $d_{\pi_{min}} = -1$ *for inverse quadratic, Gaussian and inverse multiquadratic RBFs,*

- $d_{\pi_{min}} = 0$ *for the multiquadratic and linear RBFs,*
- $d_{\pi_{min}} = 1$ *for the thin plate spline RBF,*

ensures that the linear system of equations in (2.31) *admits a unique solution [54, 107].*

As a final remark, note that, under the assumptions of Proposition 2.1, $\hat{f}_N\left(\boldsymbol{x}; \boldsymbol{\alpha}_f, \boldsymbol{\beta}_f, \epsilon_f\right)$ in (2.29) is still differentiable everywhere with respect to $\boldsymbol{x}$, since polynomials are differentiable everywhere.

2.5.2 Gaussian process regression

Differently from the surrogate models in (2.20) and (2.29), which are deterministic, a Gaussian Process ($\mathcal{GP}$) [153] is a probabilistic model that describes the distribution of the values that the cost function $f(\boldsymbol{x})$ of the GOP (2.1) can assume. A $\mathcal{GP}$ is a generalization of the Gaussian probability distribution and is defined as follows.

Definition 2.8: Gaussian process [153]. *A Gaussian process is a collection of random variables, any finite number of which have a joint Gaussian distribution.*

A $\mathcal{GP}$ is completely specified by its mean function, $\mu_f : \mathbb{R}^n \to \mathbb{R}$, and covariance function (or kernel), $k_f : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$:

$$\mu_f(\boldsymbol{x}) = \mathbb{E}\left[f(\boldsymbol{x})\right], \tag{2.32}$$

$$k_f\left(\boldsymbol{x}, \tilde{\boldsymbol{x}}; \boldsymbol{\theta}_f\right) = \mathbb{E}\left[\left(f(\boldsymbol{x}) - \mu_f(\boldsymbol{x})\right) \cdot \left(f(\tilde{\boldsymbol{x}}) - \mu_f(\tilde{\boldsymbol{x}})\right)\right], \tag{2.33}$$

where $\boldsymbol{\theta}_f$ is a vector of hyper-parameters for the kernel (similarly to the shape parameter for the RBFs in (2.19)). For convenience, we assume that $\mu_f(\boldsymbol{x})$ is the zero function, i.e. $\mu_f(\boldsymbol{x}) = 0$. Notation-wise, whenever we model $f(\boldsymbol{x})$ using a Gaussian process, we denote it as:

$$f(\boldsymbol{x}) \sim \mathcal{GP}\left(0, k_f\left(\boldsymbol{x}, \tilde{\boldsymbol{x}}; \boldsymbol{\theta}_f\right)\right). \tag{2.34}$$

In the context of black-box optimization, we assume to have at our disposal the set of samples $\mathcal{X}$ in (2.9) and the measures of the cost function $\mathcal{Y}$ in (2.10). In the $\mathcal{GP}$ setting, differently from Assumption 2.4, we suppose that the measures of $f(\boldsymbol{x})$ are affected by Gaussian noise as follows.

Assumption 2.5. *The measures of the cost function $f(\boldsymbol{x})$ are affected by zero-mean Gaussian noise with variance $\sigma_f^2 \in \mathbb{R}_{\geq 0}$, i.e.:*

$$y_i = f(\boldsymbol{x}_i) + \eta_{f_i}, \quad \eta_{f_i} \overset{i.i.d.}{\sim} \mathcal{N}\left(0, \sigma_f^2\right), \forall i = 1, \ldots, N, \boldsymbol{x}_i \in \mathcal{X}, y_i \in \mathcal{Y}. \tag{2.35}$$

In addition to the vector of measures $\boldsymbol{y}$ in (2.27), we also consider the vector containing the noiseless values of $f(\cdot)$, i.e.:

$$\boldsymbol{f} = \begin{bmatrix} f(\boldsymbol{x}_1) & \dots & f(\boldsymbol{x}_N) \end{bmatrix}^\top \in \mathbb{R}^N. \tag{2.36}$$

Black-box optimization methods which use surrogates based on Gaussian processes follow the traditional *Bayesian learning paradigm* [77]. In particular, an approximation of $f(\boldsymbol{x})$ of the GOP (2.1) is obtained through a process called Gaussian process regression. First of all, we must choose a suitable *prior distribution* for the cost function $f(\boldsymbol{x})$ which, in this case, is the one in (2.34). In practice, we need to select a suitable kernel function $k_f(\boldsymbol{x}, \tilde{\boldsymbol{x}}; \boldsymbol{\theta}_f)$; the following Theorem defines which functions can be used as kernels.

Theorem 2.2: Mercer's Theorem [30]

Let $k_f : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ be a symmetric[a] function and consider the set of samples $\mathcal{X}$ in (2.9). $k_f(\boldsymbol{x}, \tilde{\boldsymbol{x}}; \boldsymbol{\theta}_f)$ is a kernel function if and only if the matrix

$$K_f(\boldsymbol{\theta}_f) = \begin{bmatrix} k_f(\boldsymbol{x}_1, \boldsymbol{x}_1; \boldsymbol{\theta}_f) & \cdots & k_f(\boldsymbol{x}_1, \boldsymbol{x}_N; \boldsymbol{\theta}_f) \\ \vdots & \ddots & \vdots \\ k_f(\boldsymbol{x}_N, \boldsymbol{x}_1; \boldsymbol{\theta}_f) & \cdots & k_f(\boldsymbol{x}_N, \boldsymbol{x}_N; \boldsymbol{\theta}_f) \end{bmatrix}, \tag{2.37}$$

whose (i,j)-th entry is $K_f^{(i,j)}(\boldsymbol{\theta}_f) = k_f(\boldsymbol{x}_i, \boldsymbol{x}_j; \boldsymbol{\theta}_f)$, is positive semidefinite. We refer to $K_f(\boldsymbol{\theta}_f)$ in (2.37) as the Gram matrix.

[a]A function $k_f : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ is said to be symmetric if and only if $k_f(\boldsymbol{x}_1, \boldsymbol{x}_2) = k_f(\boldsymbol{x}_2, \boldsymbol{x}_1), \forall \boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathbb{R}^n$

In practice, some of the radial basis functions in Definition 2.6 can be used as kernels[3]. A popular one is the *squared exponential kernel* [153]:

$$k_f(\boldsymbol{x}, \tilde{\boldsymbol{x}}) = \exp\left\{-\frac{1}{2} \cdot \|\boldsymbol{x} - \tilde{\boldsymbol{x}}\|_2^2\right\},$$

which is none other than a Gaussian RBF with shape parameter $\epsilon_f = \frac{1}{\sqrt{2}}$. Alternatively, the squared exponential kernel can also include two hyper-parameters:

$$k_f(\boldsymbol{x}, \tilde{\boldsymbol{x}}; \boldsymbol{\theta}_f) = \left(\theta_f^{(1)}\right)^2 \cdot \exp\left\{-\frac{1}{2} \cdot \left(\frac{\|\boldsymbol{x} - \tilde{\boldsymbol{x}}\|_2}{\theta_f^{(2)}}\right)^2\right\},$$

where $\theta_f^{(1)}$ is referred to as the *signal standard deviation* whereas $\theta_f^{(2)}$ is the *characteristic length scale*.

[3]When that is the case, the Gram matrix in (2.37) is equivalent to $\Phi_f(\epsilon_f)$ in (2.26).

Now, we go back to the problem of finding a surrogate model for the cost function following the Bayesian learning paradigm. By imposing the $\mathcal{GP}$ prior in (2.34) on the cost function $f(\boldsymbol{x})$, the probability distribution of $\boldsymbol{f}$ in (2.36) is:

$$p\left(\boldsymbol{f};\boldsymbol{\theta}_f\right) = \mathcal{N}\left(\boldsymbol{\mu}_f, \Sigma_f\right), \tag{2.38a}$$

$$\boldsymbol{\mu}_f = \mathbf{0}_N, \tag{2.38b}$$

$$\Sigma_f = K_f\left(\boldsymbol{\theta}_f\right). \tag{2.38c}$$

Then, from (2.35), the *likelihood* $p\left(\boldsymbol{y}\,\middle|\,\boldsymbol{f},\mathcal{X}\right)$ is also Gaussian and it is equal to:

$$p\left(\boldsymbol{y}\,\middle|\,\boldsymbol{f},\mathcal{X}\right) = \mathcal{N}\left(\boldsymbol{\mu}_{\boldsymbol{y}\,\middle|\,\boldsymbol{f},\mathcal{X}}, \Sigma_{\boldsymbol{y}\,\middle|\,\boldsymbol{f},\mathcal{X}}\right), \tag{2.39a}$$

$$\boldsymbol{\mu}_{\boldsymbol{y}\,\middle|\,\boldsymbol{f},\mathcal{X}} = \boldsymbol{f}, \tag{2.39b}$$

$$\Sigma_{\boldsymbol{y}\,\middle|\,\boldsymbol{f},\mathcal{X}} = \sigma_f^2 \cdot I_N. \tag{2.39c}$$

Similarly, the *marginal likelihood* $p\left(\boldsymbol{y}\,\middle|\,\mathcal{X};\boldsymbol{\theta}_f\right)$ is Gaussian since (2.35) is the sum of two normally distributed random variables, in particular:

$$p\left(\boldsymbol{y}\,\middle|\,\mathcal{X};\boldsymbol{\theta}_f\right) = \mathcal{N}\left(\boldsymbol{\mu}_{\boldsymbol{y}\,\middle|\,\mathcal{X}}, \Sigma_{\boldsymbol{y}\,\middle|\,\mathcal{X}}\right), \tag{2.40a}$$

$$\boldsymbol{\mu}_{\boldsymbol{y}\,\middle|\,\mathcal{X}} = \mathbf{0}_N, \tag{2.40b}$$

$$\Sigma_{\boldsymbol{y}\,\middle|\,\mathcal{X}} = K_f\left(\boldsymbol{\theta}_f\right) + \sigma_f^2 \cdot I_N. \tag{2.40c}$$

On a side note, the marginal likelihood in (2.40) is typically used to calibrate the hyper-parameters $\boldsymbol{\theta}_f$ of the kernel by solving the following optimization problem [153]:

$$\begin{aligned} &\arg\min_{\boldsymbol{\theta}_f} \left[-\ln p\left(\boldsymbol{y}\,\middle|\,\mathcal{X};\boldsymbol{\theta}_f\right)\right] \\ &\text{s.t.} \quad \boldsymbol{\theta}_f \in \Theta, \end{aligned} \tag{2.41}$$

where Θ is a set of constraints for $\boldsymbol{\theta}_f$ (such as the non-negativity of some hyper-parameters). Typically, the variance σ_f^2 of the Gaussian noise in (2.35) is also estimated via Problem (2.41). This way of estimating the hyper-parameters of the kernel is commonly referred to as *Maximum Likelihood Estimation (MLE)*.

Next, we apply Bayes' theorem using the distributions in (2.38), (2.39) and (2.40):

$$p\left(\boldsymbol{f}\,\middle|\,\boldsymbol{y},\mathcal{X};\boldsymbol{\theta}_f\right) = \frac{p\left(\boldsymbol{y}\,\middle|\,\boldsymbol{f},\mathcal{X}\right) \cdot p\left(\boldsymbol{f};\boldsymbol{\theta}_f\right)}{p\left(\boldsymbol{y}\,\middle|\,\mathcal{X};\boldsymbol{\theta}_f\right)}, \tag{2.42}$$

to obtain the *posterior distribution* $p\left(\boldsymbol{f} \,\middle|\, \boldsymbol{y}, \mathcal{X}; \boldsymbol{\theta}_f\right)$. Using the conjugacy properties of the Gaussian distribution [15], it is possible to prove that:

$$p\left(\boldsymbol{f} \,\middle|\, \boldsymbol{y}, \mathcal{X}; \boldsymbol{\theta}_f\right) = \mathcal{N}\left(\boldsymbol{\mu}_{\boldsymbol{f}\left|\boldsymbol{y},\mathcal{X}\right.}, \Sigma_{\boldsymbol{f}\left|\boldsymbol{y},\mathcal{X}\right.}\right), \tag{2.43a}$$

$$\boldsymbol{\mu}_{\boldsymbol{f}\left|\boldsymbol{y},\mathcal{X}\right.} = K_f\left(\boldsymbol{\theta}_f\right) \cdot \left[K_f\left(\boldsymbol{\theta}_f\right) + \sigma_f^2 \cdot I_N\right]^{-1} \cdot \boldsymbol{y}, \tag{2.43b}$$

$$\Sigma_{\boldsymbol{f}\left|\boldsymbol{y},\mathcal{X}\right.} = \sigma_f^2 \cdot K_f\left(\boldsymbol{\theta}_f\right) \cdot \left[K_f\left(\boldsymbol{\theta}_f\right) + \sigma_f^2 \cdot I_N\right]^{-1}. \tag{2.43c}$$

Note that, by Mercer's Theorem 2.2, $K_f\left(\boldsymbol{\theta}_f\right)$ is positive semidefinite. Hence, $\left[K_f\left(\boldsymbol{\theta}_f\right) + \sigma_f^2 \cdot I_N\right]$ in (2.43) is invertible $\forall \sigma_f^2 \in \mathbb{R}_{>0}$.

In practice, we are actually interested in predicting the value of the cost function $f(\boldsymbol{x})$ at a point that has not been previously evaluated, i.e. we want to estimate $\tilde{f} = f(\tilde{\boldsymbol{x}})$ for some $\tilde{\boldsymbol{x}} \notin \mathcal{X}$ in (2.9). To do so, we define the prior on the random vector $\begin{bmatrix}\boldsymbol{f}^\top & \tilde{f}\end{bmatrix}^\top$ analogously to (2.38). Then, we find the joint posterior distribution $p\left(\boldsymbol{f}, \tilde{f} \,\middle|\, \boldsymbol{y}, \mathcal{X}, \tilde{\boldsymbol{x}}; \boldsymbol{\theta}_f\right)$ following the same rationale used to find $p\left(\boldsymbol{f} \,\middle|\, \boldsymbol{y}, \mathcal{X}; \boldsymbol{\theta}_f\right)$ in (2.43). Lastly, we marginalize with respect to $\tilde{f}$, obtaining the *predictive distribution* [153]:

$$p\left(\tilde{f} \,\middle|\, \boldsymbol{f}, \boldsymbol{y}, \mathcal{X}, \tilde{\boldsymbol{x}}; \boldsymbol{\theta}_f\right) = \mathcal{N}\left(\mu_{\tilde{f}\left|\boldsymbol{f},\boldsymbol{y},\mathcal{X},\tilde{\boldsymbol{x}}\right.}, \Sigma_{\tilde{f}\left|\boldsymbol{f},\boldsymbol{y},\mathcal{X},\tilde{\boldsymbol{x}}\right.}\right), \tag{2.44a}$$

$$\mu_{\tilde{f}\left|\boldsymbol{f},\boldsymbol{y},\mathcal{X},\tilde{\boldsymbol{x}}\right.} = \boldsymbol{k}_f\left(\tilde{\boldsymbol{x}}; \boldsymbol{\theta}_f\right)^\top \cdot \left[K_f\left(\boldsymbol{\theta}_f\right) + \sigma_f^2 \cdot I_N\right]^{-1} \cdot \boldsymbol{y}, \tag{2.44b}$$

$$\Sigma_{\tilde{f}\left|\boldsymbol{f},\boldsymbol{y},\mathcal{X},\tilde{\boldsymbol{x}}\right.} = k_f\left(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{x}}; \boldsymbol{\theta}_f\right) - \boldsymbol{k}_f\left(\tilde{\boldsymbol{x}}; \boldsymbol{\theta}_f\right)^\top \cdot \left[K_f\left(\boldsymbol{\theta}_f\right) + \sigma_f^2 \cdot I_N\right]^{-1} \cdot \boldsymbol{k}_f\left(\tilde{\boldsymbol{x}}; \boldsymbol{\theta}_f\right), \tag{2.44c}$$

where $\boldsymbol{k}_f\left(\tilde{\boldsymbol{x}}; \boldsymbol{\theta}_f\right) \in \mathbb{R}^N$ is the *kernel vector*:

$$\boldsymbol{k}_f\left(\tilde{\boldsymbol{x}}; \boldsymbol{\theta}_f\right) = \begin{bmatrix} k_f\left(\boldsymbol{x}_1, \tilde{\boldsymbol{x}}; \boldsymbol{\theta}_f\right) & \dots & k_f\left(\boldsymbol{x}_N, \tilde{\boldsymbol{x}}; \boldsymbol{\theta}_f\right)\end{bmatrix}^\top. \tag{2.45}$$

Finally, the surrogate model of the cost function $f(\boldsymbol{x})$ of the GOP (2.1) is the mean of the predictive distribution in (2.44):

$$\hat{f}_N\left(\boldsymbol{x}; \boldsymbol{\beta}_f, \boldsymbol{\theta}_f\right) = \boldsymbol{k}_f\left(\boldsymbol{x}; \boldsymbol{\theta}_f\right)^\top \cdot \boldsymbol{\beta}_f, \tag{2.46}$$

where $\boldsymbol{\beta}_f = \left[K_f\left(\boldsymbol{\theta}_f\right) + \sigma_f^2 \cdot I_N\right]^{-1} \cdot \boldsymbol{y}$.

Remark 2.3 (Relationship to the RBF expansion surrogate model)**.** *The surrogate model in* (2.46) *resulting from Assumption 2.5 is quite similar to the radial basis function expansion in* (2.20)*. Notably, in the noiseless case (*$\sigma_f^2 = 0$*), the weight vector* $\boldsymbol{\beta}_f$ *in* (2.46) *is computed as* $\boldsymbol{\beta}_f = K_f\left(\boldsymbol{\theta}_f\right)^{-1} \cdot \boldsymbol{y}$*, which is equivalent to solving the linear system in* (2.25)*, provided that the kernel function is a suitable radial basis function (such as the ones highlighted in Proposition 2.2).*

2.6 Infill sampling criteria

An infill sampling criterion (Step 7 of Algorithm 5) of a surrogate-based method is a strategy for selecting new candidate samples for evaluation. Typically, only one point is selected at each iteration of Algorithm 5. Consider the unconstrained black-box optimization problem ($\Xi = \mathbb{R}^n$) in (2.1) and suppose to have at our disposal N samples in $\mathcal{X}$ (2.9). The simplest way to select a *new candidate sample* $\boldsymbol{x}_{N+1} \in \Omega$ is to solve the following global optimization problem (*pure exploitation*):

$$\begin{aligned} \boldsymbol{x}_{N+1} &= \arg \min_{\boldsymbol{x}} \hat{f}_N(\boldsymbol{x}) \\ \text{s.t.} \quad & \boldsymbol{x} \in \Omega. \end{aligned} \tag{2.47}$$

However, as pointed out in [149], any surrogate-based method which uses the infill sampling criterion described by Problem (2.47) is not guaranteed to converge to a global (or even local) minimizer of the GOP (2.1). Instead, most response surface techniques rely on a so-called acquisition function, $a_N : \mathbb{R}^n \to \mathbb{R}$, which trades off exploration and exploitation. The new candidate sample is selected by solving either[4]

$$\begin{aligned} \boldsymbol{x}_{N+1} &= \arg \min_{\boldsymbol{x}} a_N(\boldsymbol{x}) && \text{(2.48a)} \\ \text{s.t.} \quad & \boldsymbol{x} \in \Omega \quad \text{or} \\ \boldsymbol{x}_{N+1} &= \arg \max_{\boldsymbol{x}} a_N(\boldsymbol{x}) && \text{(2.48b)} \\ \text{s.t.} \quad & \boldsymbol{x} \in \Omega, \end{aligned}$$

depending on definition of the acquisition function. In the context of constrained BBO, the black-box constraints in Ξ can be handled by modifying $a_N(\boldsymbol{x})$ so that the exploration of those regions of Ω that are likely to contain Ξ-infeasible samples is penalized. As we will see shortly, there also exist several alternatives to the strategy in (2.48).

Many response surface techniques share the same surrogate models (such as the ones reviewed in Section 2.5) but use different infill sampling criteria. In what follows, we distinguish between surrogate-based methods that use deterministic surrogates (such as the RBF expansion in Section 2.5.1) and response surface techniques that, instead, rely on probabilistic models (e.g. based on $\mathcal{GP}$s as in Section 2.5.2). Furthermore, consistently with [93], we refer to the latter as Bayesian Optimization (BayesOpt) procedures. In the next Sections, we review a plethora of infill sampling criteria used by the most popular black-box optimization algorithms.

In what follows, we assume to have performed $N \in \mathbb{N}$ sample evaluations. Hence, sets $\mathcal{X}$ in (2.9), $\mathcal{Y}$ in (2.10), $\mathcal{U}_\Xi$ in (2.11) and (possibly but not necessarily) $\mathcal{C}_\Xi$ in (2.12) are available. Moreover, we

[4]Problems (2.48a) and (2.48b) could admit multiple global solutions but we are interested in just one of them.

refrain from reporting the hyper-parameters of the surrogate models, such as ϵ_f for (2.20) and (2.29), as well as $\boldsymbol{\theta}_f$ for (2.44), to ease the notation.

2.6.1 Infill sampling criteria for methods based on deterministic surrogates

Most methods that use deterministic surrogates presume that Assumption 2.4 holds and behave accordingly.

Unconstrained black-box optimization

We start our review by considering the unconstrained black-box optimization framework ($\Xi = \mathbb{R}^n$).

`Gutmann-RBF` [54]. The method proposed in [54] uses the surrogate model in (2.29) and an acquisition function that is a measure of "bumpiness" for $\hat{f}_N(\boldsymbol{x})$. The rationale behind it is the following. At each iteration of the `Gutmann-RBF` [54] algorithm, a target value $\tau \in (-\infty, \min_{\boldsymbol{x}\in\Omega} \hat{f}_N(\boldsymbol{x})]$[5] for which to aim for is chosen; τ can be seen as an estimate of the global minimum of the GOP (2.1). The new candidate sample $\boldsymbol{x}_{N+1} \notin \mathcal{X}$ is selected such that the surrogate model $\hat{f}_{N+1}(\boldsymbol{x})$ in (2.29) that we would obtain by interpolating the sample evaluations in $\mathcal{X} \cup \{\boldsymbol{x}_{N+1}\}$ and $\mathcal{Y} \cup \{\tau\}$ is the most smooth (i.e. the least "bumpy"), see Figure 7 for an example. In [54], the author shows that there exists a

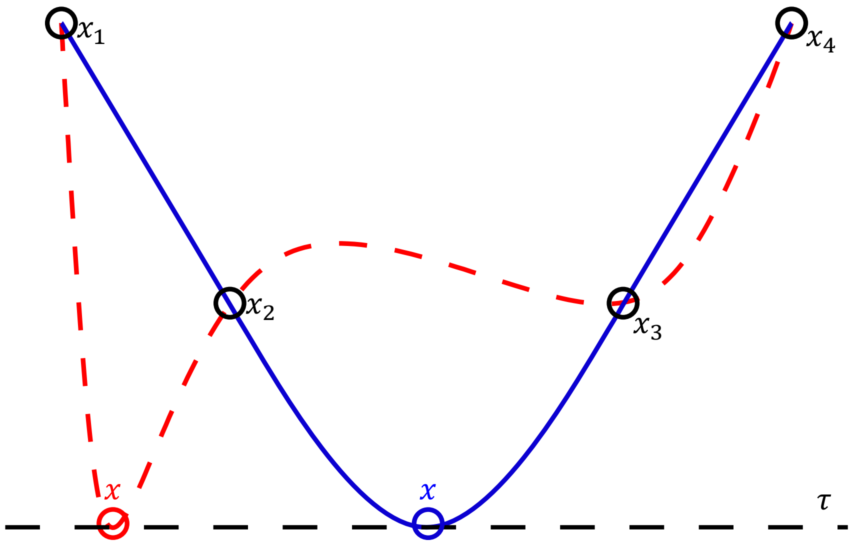


Figure 7: One-dimensional example, taken from [27], which highlights what it means for a surrogate to be "bumpy" for the `Gutmann-RBF` [54] algorithm. Suppose to have evaluated four samples, $\mathcal{X} = \{x_1, x_2, x_3, x_4\}$ (black circles), and fix a target value τ. The Figure displays two surrogate models (blue and dashed red lines) obtained by interpolating both the previously evaluated samples and a new one, x, at two different locations. The `Gutmann-RBF` [54] method assumes that it is more likely that a point with cost τ is located at the blue circle rather than at the red one because the resulting interpolant is less "bumpy".

[5]The case $\tau = \min_{\boldsymbol{x}\in\Omega} \hat{f}_N(\boldsymbol{x})$ is admissible only if $\min_{\boldsymbol{x}\in\Omega} \hat{f}_N(\boldsymbol{x}) < y_i, \forall y_i \in \mathcal{Y}$.

natural measure of "bumpiness" $\tilde{a}_N(\boldsymbol{x})$ for $\hat{f}_N(\cdot)$ in (2.29), which can be computed for any $\boldsymbol{x} \in \mathbb{R}^n$ that has not been previously evaluated, namely:

$$\tilde{a}_N(\boldsymbol{x}) = (-1)^{\left(d_{\pi_{min}}+1\right)} \cdot \zeta(\boldsymbol{x}) \cdot \left[\tau - \hat{f}_N(\boldsymbol{x})\right]^2, \quad \forall \boldsymbol{x} \in \mathbb{R}^n \setminus \mathcal{X}. \tag{2.49}$$

$d_{\pi_{min}}$ in (2.49) is set as in Proposition 2.3 and $\zeta(\boldsymbol{x})$ is obtained by solving the following linear system:

$$\begin{bmatrix} \Phi_f(\epsilon_f) & \boldsymbol{\phi}_f(\boldsymbol{x};\epsilon_f) & P_f \\ \boldsymbol{\phi}_f(\boldsymbol{x};\epsilon_f)^\top & \varphi_f(0) & \boldsymbol{\pi}_f(\boldsymbol{x})^\top \\ P_f^\top & \boldsymbol{\pi}_f(\boldsymbol{x}) & 0_{N_\pi \times N_\pi} \end{bmatrix} \cdot \begin{bmatrix} \boldsymbol{\beta}_f(\boldsymbol{x}) \\ \zeta(\boldsymbol{x}) \\ \boldsymbol{\alpha}_f(\boldsymbol{x}) \end{bmatrix} = \begin{bmatrix} \boldsymbol{0}_N \\ 1 \\ \boldsymbol{0}_{N_\pi} \end{bmatrix},$$

which can be seen as imposing the interpolation conditions in (2.31) to a set of samples of the decision vector and measures of the cost function defined by the pairs $(\boldsymbol{x}_1, 0), \ldots, (\boldsymbol{x}_N, 0), (\boldsymbol{x}, 1)$. In particular, in [16], the authors propose an efficient way to compute $\zeta(\boldsymbol{x})$ that is based on a factorization of the interpolation matrix. A better alternative to the minimization of $\tilde{a}_N(\boldsymbol{x})$ in (2.49) (which is not defined at each $\boldsymbol{x}_i \in \mathcal{X}$) is the maximization of the following acquisition function:

$$a_N(\boldsymbol{x}) = \begin{cases} \frac{1}{\tilde{a}_N(\boldsymbol{x})} & \text{if } \boldsymbol{x} \in \mathbb{R}^n \setminus \mathcal{X} \\ 0 & \text{otherwise} \end{cases}, \tag{2.50}$$

as in Problem (2.48b). The choice of the target value τ in (2.49) is critical: targets that are close to $\min_{\boldsymbol{x} \in \Omega} \hat{f}_N(\boldsymbol{x})$ make the search for $\boldsymbol{x}_{N+1}$ highly local (exploitation), whereas setting $\tau = -\infty$ leads to a more exploratory behavior. In practice, the target values are cycled in between the iterations of the `Gutmann-RBF` [54] algorithm, following a sequence $\mathcal{T}_{cycle} = \langle \tau_0, \ldots, \tau_{N_{cycle}-1} \rangle$ that is repeated every $N_{cycle} \in \mathbb{N}$ iterations. Furthermore, in [54], it is proven that by choosing the target values in a particular manner (such as by including a term $\tau_j = -\infty$ in $\mathcal{T}_{cycle}$) and under suitable choices of $\varphi_f(\cdot)$ and d_π for the surrogate model in (2.29), the method converges to the global minimum of any continuous function over some compact subset of $\mathbb{R}^n$.

`MSRS` [116]. The Metric Stochastic Response Surface method (`MSRS` [116]) is a black-box optimization procedure that can be used for bound constrained optimization problems (i.e. with Ω as in (1.3)). We focus on an implementation of the method based on the surrogate $\hat{f}_N(\boldsymbol{x})$ in (2.29), which is referred to as `MSRBF` [116]. `MSRBF` [116] uses the following exploration function:

$$z_N(\boldsymbol{x}) = -\min_{\boldsymbol{x}_i \in \mathcal{X}} \|\boldsymbol{x} - \boldsymbol{x}_i\|_2, \tag{2.51}$$

as well as the surrogate $\hat{f}_N(\boldsymbol{x})$, to drive the search for new candidate samples. $z_N(\boldsymbol{x})$ in (2.51) is simply the (negative) distance of a point $\boldsymbol{x} \in \mathbb{R}^n$ from its closest sample in $\mathcal{X}$. No additional global optimization problem is solved when looking for $\boldsymbol{x}_{N+1}$. Instead, at each iteration, the `MSRBF` [116] procedure randomly generates a set of samples $\mathcal{X}_\Omega$ using either a uniform random distribution on Ω (Global `MSRBF` [116]) or by adding random perturbations to the current best solution $\boldsymbol{x_{best}}(N)$ (Local `MSRBF` [116]); then, $\boldsymbol{x}_{N+1}$ is chosen among the points contained inside $\mathcal{X}_\Omega$. Consider the minimum and maximum of $\hat{f}_N(\boldsymbol{x})$ and $z_N(\boldsymbol{x})$ over $\mathcal{X}_\Omega$, which we denote as $\hat{f}_{N_{min}}(\mathcal{X}_\Omega) = \min_{\boldsymbol{x}\in\mathcal{X}_\Omega} \hat{f}_N(\boldsymbol{x}), \hat{f}_{N_{max}}(\mathcal{X}_\Omega) = \max_{\boldsymbol{x}\in\mathcal{X}_\Omega} \hat{f}_N(\boldsymbol{x})$ and analogously for $z_N(\boldsymbol{x})$. The new candidate sample is selected among the points in $\mathcal{X}_\Omega$ as the one that minimizes the following acquisition function:

$$a_N(\boldsymbol{x}) = \delta \cdot \bar{\hat{f}}_N(\boldsymbol{x};\mathcal{X}_\Omega) + (1-\delta)\cdot \bar{z}_N(\boldsymbol{x};\mathcal{X}_\Omega), \tag{2.52}$$

where $\delta \in [0,1]$ is the exploration-exploitation trade-off weight and

$$\bar{\hat{f}}_N(\boldsymbol{x};\mathcal{X}_\Omega) = \begin{cases} \frac{\hat{f}_N(\boldsymbol{x})-\hat{f}_{N_{min}}(\mathcal{X}_\Omega)}{\hat{f}_{N_{max}}(\mathcal{X}_\Omega)-\hat{f}_{N_{min}}(\mathcal{X}_\Omega)} & \text{if } \hat{f}_{N_{max}}(\mathcal{X}_\Omega) \neq \hat{f}_{N_{min}}(\mathcal{X}_\Omega) \\ 1 & \text{otherwise} \end{cases},$$

$$\bar{z}_N(\boldsymbol{x};\mathcal{X}_\Omega) = \begin{cases} \frac{z_N(\boldsymbol{x})-z_{N_{max}}(\mathcal{X}_\Omega)}{z_{N_{max}}(\mathcal{X}_\Omega)-z_{N_{min}}(\mathcal{X}_\Omega)} & \text{if } z_{N_{max}}(\mathcal{X}_\Omega) \neq z_{N_{min}}(\mathcal{X}_\Omega) \\ 1 & \text{otherwise} \end{cases}.$$

Hence, the new candidate sample is chosen as:

$$\boldsymbol{x}_{N+1} = \arg\min_{\boldsymbol{x}\in\mathcal{X}_\Omega} a_N(\boldsymbol{x}). \tag{2.53}$$

Similarly to `Gutmann-RBF` [54], the exploration-exploitation trade-off weight δ of algorithm `MSRBF` [116] is varied in between iterations, following a sequence $\Delta_{cycle} = \langle \delta_0, \dots, \delta_{N_{cycle}-1}\rangle$ that is repeated every $N_{cycle} \in \mathbb{N}$ iterations. Furthermore, the authors propose to use the sequence $\Delta_{cycle} = \langle 0.2, 0.4, 0.6, 0.9, 0.95, 1\rangle$ as to properly alternate between local and global search. Under some mild assumptions on how the set $\mathcal{X}_\Omega$ is generated, `MSRS` [116] is shown to be convergent almost surely. An extension of `MSRS` [116] that is more suited for higher-dimensional problems, called `SO-SA`, has been proposed in [151]. Therein, the authors suggest to draw δ in (2.52) from a uniform random distribution $\mathcal{U}(0,1)$ and keep the trade-off unaltered until it fails to find a significantly better solution (greedy approach). `SO-SA` [151] also revisits the generation of the set $\mathcal{X}_\Omega$ to take into consideration local sensitivity information.

CORS [115]. Constrained Optimization using Response Surfaces (`CORS` [115]) is an algorithm proposed by the same authors of `MSRS` [116] that can handle nonlinearly constrained global optimization problems, i.e. with Ω defined as in (1.5). In `CORS` [115], the new candidate sample for evaluation is chosen as the point that minimizes a surrogate of the cost function over Ω, but that is sufficiently far away from the samples in $\mathcal{X}$ (2.9). In [115], the authors consider $\hat{f}_N(\boldsymbol{x})$ as defined in (2.29), giving rise to `CORS-RBF`. The infill sampling criterion for the `CORS` [115] algorithm is:

$$\begin{aligned} \boldsymbol{x}_{N+1} = \arg\min_{\boldsymbol{x}} & \hat{f}_N(\boldsymbol{x}) \\ \text{s.t.} \quad & \boldsymbol{x} \in \Omega \\ & \|\boldsymbol{x} - \boldsymbol{x}_i\|_2 \geq \delta \cdot d_\Omega(\mathcal{X}) \qquad i = 1, \dots, N, \boldsymbol{x}_i \in \mathcal{X}, \end{aligned} \tag{2.54}$$

where

$$d_\Omega(\mathcal{X}) = \max_{\boldsymbol{x}\in\Omega} \min_{\boldsymbol{x}_i\in\mathcal{X}} \|\boldsymbol{x} - \boldsymbol{x}_i\|_2 \tag{2.55}$$

is a limit on how far $\boldsymbol{x}_{N+1}$ can be from the previously evaluated points, and $\delta \in [0, 1]$ is the exploration-exploitation trade-off parameter. In particular, setting $\delta = 0$ results in pure exploitation whereas higher values of δ lead to global search. Like the previously mentioned methods, the trade-off parameter δ for `CORS` [115] is cycled in between iterations of the procedure, following a sequence $\Delta_{cycle} = \langle \delta_0, \dots, \delta_{N_{cycle}-1} \rangle$ that is repeated every $N_{cycle} \in \mathbb{N}$ iterations. In particular, if Δ_{cycle} includes at least one non-zero entry, then the procedure converges to the global minimum of any continuous function over some compact set.

`GLIS` [10]. GLobal minimum using Inverse distance weighting and Surrogate radial basis functions (`GLIS` [10]) is a more recent black-box optimization method which uses the surrogate model $\hat{f}_N(\boldsymbol{x})$ in (2.20). The linear system in (2.25) is solved using a low-rank approximation of $\Phi_f(\epsilon_f)$ in (2.26) to avoid issues related to the condition number of the interpolation matrix. `GLIS` [10] looks for new candidate samples by trading off the exploitation of the surrogate model $\hat{f}_N(\boldsymbol{x})$ and the exploratory contributions brought by two exploration functions, $z_N : \mathbb{R}^n \to (-1, 0]$ and $s_N : \mathbb{R}^n \to \mathbb{R}$, which are defined as:

$$z_N(\boldsymbol{x}) = \begin{cases} 0 & \text{if } \boldsymbol{x} \in \mathcal{X} \\ -\frac{2}{\pi} \cdot \arctan\left(\frac{1}{\sum_{i=1}^N w_i(\boldsymbol{x})}\right) & \text{otherwise} \end{cases}, \tag{2.56a}$$

$$s_N(\boldsymbol{x}) = -\sqrt{\sum_{i=1}^N v_i(\boldsymbol{x}) \cdot \left(y_i - \hat{f}_N(\boldsymbol{x})\right)^2}, \tag{2.56b}$$

where:

$$v_i(\boldsymbol{x}) = \begin{cases} 1 & \text{if } \boldsymbol{x} = \boldsymbol{x}_i, \boldsymbol{x}_i \in \mathcal{X} \\ 0 & \text{if } \boldsymbol{x} = \boldsymbol{x}_j, j \neq i, \boldsymbol{x}_i, \boldsymbol{x}_j \in \mathcal{X} \\ \frac{\tilde{w}_i(\boldsymbol{x})}{\sum_{j=1}^N \tilde{w}_j(\boldsymbol{x})} & \text{otherwise} \end{cases}. \tag{2.57}$$

The exploration functions in (2.56) are based on the Inverse Distance Weighting (IDW) functions $w_i : \mathbb{R}^n \setminus \{\boldsymbol{x}_i\} \rightarrow \mathbb{R}_{>0}$, $w_i(\boldsymbol{x}) = \left(\|\boldsymbol{x} - \boldsymbol{x}_i\|_2\right)^{-2}$ and $\tilde{w}_i : \mathbb{R}^n \setminus \{\boldsymbol{x}_i\} \rightarrow \mathbb{R}_{>0}$, $\tilde{w}_i(\boldsymbol{x}) = \exp\left\{-\|\boldsymbol{x} - \boldsymbol{x}_i\|_2^2\right\} \cdot \left(\|\boldsymbol{x} - \boldsymbol{x}_i\|_2\right)^{-2}$ [69, 132]. For this reason, $z_N(\boldsymbol{x})$ and $s_N(\boldsymbol{x})$ in (2.56) are referred to as IDW distance function and IDW variance function respectively. Both functions provide exploratory capabilities; the former only depends on the locations of the previously evaluated samples, while the latter takes into consideration also their corresponding measures of the cost function as well as the surrogate model. `GLIS` [10] looks for new candidate sample by minimizing the following acquisition function:

$$a_N(\boldsymbol{x}) = \hat{f}_N(\boldsymbol{x}) + \Delta Y \cdot \delta_1 \cdot z_N(\boldsymbol{x}) + \delta_2 \cdot s_N(\boldsymbol{x}), \tag{2.58}$$

as in Problem (2.48a). In (2.58), $\Delta Y = \max\left\{\max_{y_i \in \mathcal{Y}} y_i - \min_{y_i \in \mathcal{Y}} y_i, \epsilon_{\Delta Y}\right\}$ (where $\epsilon_{\Delta Y} \in \mathbb{R}_{>0}$ is a small tolerance) is used to make $z_N(\boldsymbol{x})$ comparable to the other two terms of $a_N(\boldsymbol{x})$. $\delta_1, \delta_2 \in \mathbb{R}_{\geq 0}$ are the weights for the exploration functions. At the moment, no cycling is performed on δ_1 and δ_2 and no proof of convergence is available for `GLIS` [10]. In Chapter 4, we will analyze this algorithm in greater detail. Instead, in Chapter 5, we will propose a globally convergent extension of `GLIS` [10].

Constrained black-box optimization

Now, we move onto those algorithms that can be used for constrained black-box optimization, i.e. with $\Xi \subset \mathbb{R}^n$ for the GOP (2.1) defined as in (2.2). We review two methods that are the extensions of `MSRBF` [116] and `CORS-RBF` [115] respectively. Both procedures assume that all the black-box constraints functions are measurable, i.e. the set C_Ξ in (2.12) is available. Moreover, `MSRBF` [116] and `CORS-RBF` [115] consider Ω to be defined as in (1.3) (only bound constraints).

`ConstrLMSRBF` [111]. `ConstrLMSRBF` [111] is an extension of the Local `MSRBF` [116] algorithm that is able to handle high-dimensional constrained black-box optimization problems. The method assumes that, after the initial experimental design, at least one Ξ-feasible sample is available. This time, the surrogate model in (2.29) is used to approximate both the cost function $f(\boldsymbol{x})$ and the black-box constraints functions $\boldsymbol{g}_\Xi(\boldsymbol{x})$ of the GOP (2.1). At each iteration of `ConstrLMSRBF` [111], the set of samples $\mathcal{X}_\Omega$ is generated as in Local `MSRBF` [116], i.e. by perturbing the current best candidate $\boldsymbol{x_{best}}(N)$. Then, only those points that are predicted to be Ξ-feasible by the surrogates $\hat{\boldsymbol{g}}_{\Xi_N}(\boldsymbol{x})$ are kept; thus, $\mathcal{X}_\Omega$ is replaced by:

$$\mathcal{X}_{\Omega \cap \Xi} = \left\{\boldsymbol{x} : \boldsymbol{x} \in \mathcal{X}_\Omega, \hat{\boldsymbol{g}}_{\Xi_N}(\boldsymbol{x}) \leq \boldsymbol{0}_{q_\Xi}\right\}.$$

If no points in $\mathcal{X}_\Omega$ are predicted to be Ξ-feasible, then the set $\mathcal{X}_{\Omega \cap \Xi}$ collects the ones with the minimum number of predicted constraints violations. The new candidate sample $\boldsymbol{x}_{N+1}$ is selected among the points in $\mathcal{X}_{\Omega\cap\Xi}$ (instead of $\mathcal{X}_{\Omega}$) as the one that minimizes the acquisition function in (2.52), i.e.

$$\boldsymbol{x}_{N+1} = \arg\min_{\boldsymbol{x}\in\mathcal{X}_{\Omega\cap\Xi}} a_N(\boldsymbol{x}). \tag{2.59}$$

The exploration-exploitation parameter δ in (2.52) is still cycled as proposed for `MSRBF` [116]. `ConstrLMSRBF` [111] also keeps track of the number of successful and failed iterations (in a sense that they lead to an improvement of the best candidate $\boldsymbol{x_{best}}(N)$) to modify the step size of the perturbations used to generate $\mathcal{X}_{\Omega\cap\Xi}$.

An extension of `ConstrLMSRBF` [111], which handles the case when all initial points (resulting from the experimental design) are Ξ-infeasible, is provided in [112]. The algorithm is divided into two phases, the first devoted to finding an initial Ξ-feasible sample and the second to improving the best Ξ-feasible candidate, by solving Problem (2.59). In particular, the first phase follows the same rationale of `ConstrLMSRBF` [111] but looks for $\boldsymbol{x}_{N+1}$ by minimizing a different acquisition function over $\mathcal{X}_{\Omega\cap\Xi}$:

$$a_N(\boldsymbol{x}) = \max_{1\leq j\leq q_\Xi}\left(\max\left\{\hat{g}^{(j)}_{\Xi_N}(\boldsymbol{x}), 0\right\}\right), \tag{2.60}$$

which is the maximum predicted constraint violation.

`COBRA` [112]. `COBRA` [112] is an extension of `CORS-RBF` [115] which, similarly to `ConstrLMSRBF` [111], uses the surrogate model in (2.29) for both $f(\boldsymbol{x})$ and $\boldsymbol{g}_\Xi(\boldsymbol{x})$ of the GOP (2.1). The method operates in two phases. The first phase is devoted to finding a Ξ-feasible sample (if none are available after the initial experimental design) through the following infill sampling criterion:

$$\begin{aligned}
\boldsymbol{x}_{N+1} = \arg\min_{\boldsymbol{x}}\ & a_N(\boldsymbol{x}) && && (2.61)\\
\text{s.t.}\quad & \boldsymbol{x}\in\Omega && &&\\
& \hat{g}^{(j)}_{\Xi_N}(\boldsymbol{x}) + \varsigma^{(j)}_{\Xi} \leq 0 && j = 1,\dots,q_\Xi &&\\
& \|\boldsymbol{x}-\boldsymbol{x}_i\|_2 \geq \delta\cdot d_\Omega && i = 1,\dots,N, \boldsymbol{x}_i\in\mathcal{X}, &&
\end{aligned}$$

where the acquisition function is the quadratic penalty function:

$$a_N(\boldsymbol{x}) = \sum_{j=1}^{q_\Xi}\max\left\{\hat{g}^{(j)}_{\Xi_N}(\boldsymbol{x}), 0\right\}^2. \tag{2.62}$$

d_Ω in (2.61) is defined differently from $d_\Omega(\mathcal{X})$ in (2.55) for `CORS-RBF` [115] and it is actually equal to the length of the smallest side of the box defined by Ω (regardless of $\mathcal{X}$). Similarly to the base method, $\delta\in(0,1)$ is an exploration-exploitation trade-off parameter that is cycled in between the iterations of the procedure. Lastly, $\boldsymbol{\varsigma}_\Xi\in\mathbb{R}^{q_\Xi}_{>0}$, $\boldsymbol{\varsigma}_\Xi = \begin{bmatrix}\varsigma^{(1)}_\Xi & \dots & \varsigma^{(q_\Xi)}_\Xi\end{bmatrix}$, is a vector of (small) margins that prevent sampling points that are too close to the boundary of $\hat{\Xi} = \left\{\boldsymbol{x} : \hat{\boldsymbol{g}}_{\Xi_N}(\boldsymbol{x}) \leq \boldsymbol{0}_{q_\Xi}\right\}$. Once the

`COBRA` [112] algorithm finds a Ξ-feasible sample, the procedure moves on to the second phase, where new candidate samples are sought by solving the following optimization problem:

$$\begin{aligned} \boldsymbol{x}_{N+1} = \arg\min_{\boldsymbol{x}} & \ \hat{f}_N(\boldsymbol{x}) \\ \text{s.t.} \quad & \boldsymbol{x} \in \Omega \\ & \hat{g}^{(j)}_{\Xi_N}(\boldsymbol{x}) + \varsigma^{(j)}_{\Xi} \leq 0 && j = 1, \dots, q_\Xi \\ & \|\boldsymbol{x} - \boldsymbol{x}_i\|_2 \geq \delta \cdot d_\Omega && i = 1, \dots, N, \boldsymbol{x}_i \in \mathcal{X}. \end{aligned} \tag{2.63}$$

In [112], the authors propose to use different sequences of values for δ in the two phases, i.e. for Problems (2.61) and (2.63). Furthermore, the margins $\boldsymbol{\varsigma}_\Xi$ are adjusted based on the number of subsequent successful or failed iterations (in a sense that they lead to Ξ-feasible or Ξ-infeasible samples).

2.6.2 Infill sampling criteria for Bayesian optimization methods

Bayesian optimization procedures look for new candidate samples by making use of the probabilistic interpretation of their surrogate models (see Section 2.5.2). Intuitively, the variance of the predictive distribution is a useful tool to guide the search towards those regions of Ω where $\hat{f}_N(\boldsymbol{x})$ is most uncertain. BayesOpt has gained a lot of popularity in the past two decades, after the development of the `EGO` [68] algorithm. Consequently, many different extensions and infill sampling criteria have been developed, see for example the surveys [22, 45, 131]. Here, we forego an in-depth dissertation on Bayesian optimization and instead focus only on the most commonly used acquisition functions. Furthermore, we only consider surrogate models obtained by imposing Gaussian process priors on $f(\boldsymbol{x})$ and $\boldsymbol{g}_\Xi(\boldsymbol{x})$, although other probabilistic models have been used in the past. For the remainder of this Section, we avoid the cumbersome notation of Section 2.5.2 and simply refer to the mean and the variance of the predictive distribution of $f(\boldsymbol{x})$ at a given point $\boldsymbol{x} \in \mathbb{R}^n$, i.e. $p\left(\tilde{f} \,\middle|\, \boldsymbol{f}, \boldsymbol{y}, \mathcal{X}, \boldsymbol{x}; \boldsymbol{\theta}_f\right)$ in (2.44), as $\mu_{f_N}(\boldsymbol{x})$ and $\Sigma_{f_N}(\boldsymbol{x})$ respectively. Similarly, for each black-box constraint function $g^{(j)}_\Xi(\boldsymbol{x}), j = 1, \dots, q_\Xi$, we have $\mu_{g^{(j)}_{\Xi_N}}(\boldsymbol{x})$ and $\Sigma_{g^{(j)}_{\Xi_N}}(\boldsymbol{x})$.

Unconstrained black-box optimization

We start by analyzing the unconstrained black-box optimization case ($\Xi = \mathbb{R}^n$).

Probability of improvement. In one of the earliest works on Bayesian optimization [79], the author suggests to look for new candidates by maximizing the probability of improvement:

$$\begin{aligned} a_N(\boldsymbol{x}) &= PI_N(\boldsymbol{x}) \\ &= p\left(f(\boldsymbol{x}) \leq y_{best}(N) \,\middle|\, \boldsymbol{f}, \boldsymbol{y}, \mathcal{X}, \boldsymbol{x}\right) \\ &= \Phi_{\mathcal{N}}\left(\frac{y_{best}(N) - \mu_{f_N}(\boldsymbol{x})}{\sqrt{\Sigma_{f_N}(\boldsymbol{x})}}\right), \end{aligned} \tag{2.64}$$

where $\Phi_{\mathcal{N}}(\cdot)$ is the standard normal cumulative distribution. In practice, the acquisition function in (2.64) amounts to pure exploitation [22]. A more suitable criterion for global optimization is the following:

$$\begin{aligned} a_N(\boldsymbol{x}) &= PI_{N_\delta}(\boldsymbol{x}) \\ &= p\left(f(\boldsymbol{x}) \leq y_{best}(N) - \delta \,\middle|\, \boldsymbol{f}, \boldsymbol{y}, \mathcal{X}, \tilde{\boldsymbol{x}}\right) \\ &= \Phi_{\mathcal{N}}\left(\frac{y_{best}(N) - \delta - \mu_{f_N}(\boldsymbol{x})}{\sqrt{\Sigma_{f_N}(\boldsymbol{x})}}\right), \end{aligned} \tag{2.65}$$

where $\delta \in \mathbb{R}_{\geq 0}$ is a trade-off parameter. In particular, in [79], δ is initialized to a high value, so that the algorithm prioritizes exploration in the early iterations, and gets progressively smaller to give more importance to the surrogate later on.

Expected improvement. Instead of selecting the next sample for evaluation based on the probability of it leading to an improvement, a better alternative is to use the maximum expected improvement criterion [94], which also takes into account to what degree $\boldsymbol{x}_{N+1}$ is likely to be better than the current best candidate. Formally, the improvement brought by a point $\boldsymbol{x} \in \mathbb{R}^n$ is defined as:

$$I_N(\boldsymbol{x}) = \max\left\{0, y_{best}(N) - y\right\}, \tag{2.66}$$

where $y = f(\boldsymbol{x}) + \eta_f, \eta_f \sim \mathcal{N}\left(0, \sigma_f^2\right)$, is the measure of the cost function at $\boldsymbol{x}$ (which is assumed to be corrupted by Gaussian noise, see Assumption 2.5). Then, the expected improvement is defined as:

$$\begin{aligned} a_N(\boldsymbol{x}) &= EI_N(\boldsymbol{x}) \\ &= \mathbb{E}\left[I_N(\boldsymbol{x}) \,\middle|\, \boldsymbol{f}, \boldsymbol{y}, \mathcal{X}, \boldsymbol{x}\right] \\ &= \begin{cases} \left[y_{best}(N) - \mu_{f_N}(\boldsymbol{x})\right] \cdot \Phi_{\mathcal{N}}(\tilde{z}(\boldsymbol{x})) + \sqrt{\Sigma_{f_N}(\boldsymbol{x})} \cdot \phi_{\mathcal{N}}(\tilde{z}(\boldsymbol{x})) & \text{if } \Sigma_{f_N}(\boldsymbol{x}) > 0 \\ 0 & \text{otherwise} \end{cases}, \end{aligned} \tag{2.67}$$

where:

$$\tilde{z}(\boldsymbol{x}) = \frac{y_{best}(N) - \mu_{f_N}(\boldsymbol{x})}{\sqrt{\Sigma_{f_N}(\boldsymbol{x})}}$$

and $\phi_N(\cdot)$ is the probability density function of the standard normal distribution. The new candidate sample is selected as the one that maximizes the expected improvement, as in Problem (2.48b). The first term of the acquisition function in (2.67) promotes the exploitation of $\hat{f}_N(\boldsymbol{x}) = \mu_{f_N}(\boldsymbol{x})$ in (2.46), while the second term favors the exploration in those regions of Ω where there is high uncertainty about whether or not $f(\boldsymbol{x})$ will be better than $y_{best}(N)$ [125]. $EI_N(\boldsymbol{x})$ in (2.67) is widely used due to the fact that it is the infill sampling criterion of one of the most popular Bayesian optimization procedures, namely `EGO` [68]. Many extensions of the expected improvement criterion have been developed in the past two decades, see [154] for an in-depth review. Later on, we will cover three extensions of $EI_N(\boldsymbol{x})$ in (2.67) which can be used for constrained black-box optimization.

Lower confidence bound. The last infill sampling criterion that we cover for unconstrained black-box optimization is the lower confidence bound criterion, which has been proposed in [28]. Therein, the authors look for new candidate samples by minimizing the following acquisition function:

$$\begin{aligned} a_N(\boldsymbol{x}) &= LCB_N(\boldsymbol{x}) \\ &= \mu_{f_N}(\boldsymbol{x}) - \delta \cdot \sqrt{\Sigma_{f_N}(\boldsymbol{x})}, \end{aligned} \tag{2.68}$$

where $\delta \in \mathbb{R}_{\geq 0}$ is an exploration-exploitation trade-off parameter that is chosen by the user. Intuitively, minimizing the mean of the predictive distribution, i.e. the surrogate model $\hat{f}_N(\boldsymbol{x})$ in (2.46), is pure exploitation. Instead, the minimization of $-\sqrt{\Sigma_{f_N}(\boldsymbol{x})}$ leads to finding the sample $\boldsymbol{x}_{N+1}$ which maximizes the information gain [76, 137] (pure exploration). As a matter of fact, it has been shown that a surrogate-based method which, at each iteration, looks for the next candidate sample by solving:

$$\begin{aligned} \boldsymbol{x}_{N+1} = \arg \min_{\boldsymbol{x}} \ & -\sqrt{\Sigma_{f_N}(\boldsymbol{x})} \\ \text{s.t.} \quad & \boldsymbol{x} \in \Omega \end{aligned}$$

is guaranteed to find the global minimum of the GOP (2.1), although not efficiently. Notably, under some assumptions on the cost function $f(\boldsymbol{x})$ of the GOP (2.1) (such as if the real cost function is actually drawn from the same $\mathcal{GP}$ prior used to define its surrogate model, see (2.34)), it is possible to derive lower bounds on the convergence rates of Bayesian optimization methods that rely on the $LCB_N(\boldsymbol{x})$ criterion in (2.68) [137]. To achieve such convergence rates, the parameter δ must be varied appropriately throughout the optimization procedure.

Constrained black-box optimization

Now, we move on to the constrained black-box optimization framework. We cover three different infill sampling criteria that are based on the expected improvement in (2.67).

Constrained Expected Improvement. Suppose to have at our disposal the surrogate models (predictive distributions) for both the cost function $f(\boldsymbol{x})$ and the black-box constraints functions $\boldsymbol{g}_{\Xi}(\boldsymbol{x})$ of the GOP (2.1), which are described by $\mu_{f_N}(\boldsymbol{x}), \Sigma_{f_N}(\boldsymbol{x})$ and $\mu_{g^{(j)}_{\Xi_N}}(\boldsymbol{x}), \Sigma_{g^{(j)}_{\Xi_N}}(\boldsymbol{x}), j = 1, \ldots, q_{\Xi}$, respectively. In the Bayesian framework, it is straightforward to derive the *probability of* Ξ*-feasibility* of any given sample $\boldsymbol{x} \in \mathbb{R}^n$ from the aforementioned surrogate models. In what follows, we denote the probability of Ξ-feasibility as:

$$p_N(\boldsymbol{x} \in \Xi) = p\left(\boldsymbol{x} \in \Xi \,\middle|\, \mathcal{C}_{\Xi}, \mathcal{X}, \boldsymbol{x}\right),$$

where the subscript $(\cdot)_N$ indicates that it has been estimated from the data resulting from N sample evaluations. Under the assumption that the noise terms which affect the black-box constraints are independent (see Section 2.3), we have that:

$$\begin{aligned}
p_N(\boldsymbol{x} \in \Xi) &= p\left(\boldsymbol{x} \in \Xi \,\middle|\, \mathcal{C}_{\Xi}, \mathcal{X}, \boldsymbol{x}\right) \\
&= \prod_{j=1}^{q_{\Xi}} p\left(g^{(j)}_{\Xi}(\boldsymbol{x}) \leq 0 \,\middle|\, \boldsymbol{g}^{(j)}, \boldsymbol{c}^{(j)}, \mathcal{X}, \boldsymbol{x}\right) \\
&= \prod_{j=1}^{q_{\Xi}} \Phi_N\left(-\frac{\mu_{g^{(j)}_{\Xi_N}}(\boldsymbol{x})}{\sqrt{\Sigma_{g^{(j)}_{\Xi_N}}(\boldsymbol{x})}}\right),
\end{aligned} \tag{2.69}$$

where, with a slight abuse of notation,

$$\begin{aligned}
\boldsymbol{g}^{(j)} &= \begin{bmatrix} g^{(j)}_{\Xi}(\boldsymbol{x}_1) & \ldots & g^{(j)}_{\Xi}(\boldsymbol{x}_N) \end{bmatrix}^{\top} \in \mathbb{R}^N \text{ and} \\
\boldsymbol{c}^{(j)} &= \begin{bmatrix} c^{(j)}_1 & \ldots & c^{(j)}_N \end{bmatrix}^{\top} \in \mathbb{R}^N
\end{aligned}$$

are defined similarly to $\boldsymbol{f}$ in (2.36) and $\boldsymbol{y}$ in (2.27) (cf. also the set $\mathcal{C}^{(j)}_{\Xi}$ in (2.13)). The probability of Ξ-feasibility in (2.69) can be used to penalize the search in those regions of Ω that are likely to contain Ξ-infeasible points. In particular, in [128], the author proposes to look for new candidate samples by maximizing the constrained expected improvement:

$$\begin{aligned}
a_N(\boldsymbol{x}) &= CEI_N(\boldsymbol{x}) \\
&= p_N(\boldsymbol{x} \in \Xi) \cdot EI_N(\boldsymbol{x}),
\end{aligned} \tag{2.70}$$

where $EI_N(\boldsymbol{x})$ is defined as in (2.67) although, in this case, $y_{best}(N)$ is the measure of the cost function for the best Ξ-feasible candidate. Note that if no Ξ-feasible sample has been found by the initial experimental design, then we can seek one by maximizing the probability of Ξ-feasibility in (2.69), i.e. [49]:

$$\begin{aligned}
\boldsymbol{x}_{N+1} = \arg\max_{\boldsymbol{x}} \; & p_N(\boldsymbol{x} \in \Xi) \\
\text{s.t.} \quad & \boldsymbol{x} \in \Omega,
\end{aligned} \tag{2.71}$$

resulting in a two-phase approach such as the extended `ConstrLMSRBF` [111] and `COBRA` [112] algorithms.

`SuperEGO` [125]. `SuperEGO` [125] is an extension of the popular `EGO` [68] algorithm which handles constrained black-box optimization problems through the following infill sampling criterion:

$$\begin{aligned} \boldsymbol{x}_{N+1} = \arg \max_{\boldsymbol{x}} & \, EI_N(\boldsymbol{x}) \\ \text{s.t.} \quad & \boldsymbol{x} \in \Omega \\ & p_N(\boldsymbol{x} \in \Xi) \geq \gamma, \end{aligned} \tag{2.72}$$

where $EI_N(\boldsymbol{x})$ is defined as in (2.67), $p_N(\boldsymbol{x} \in \Xi)$ as in (2.69) and $\gamma \in [0, 1]$ is a user-defined threshold (typically, $\gamma = 0.95$). In [125], the author argues that the probability of Ξ-feasibility impacts the acquisition function in (2.70) too strongly, keeping the algorithm from exploring points directly along the boundary of Ξ of the GOP (2.1) (where the true global minimizer(s) might lie). Hence, he suggests that a constrained formulation, such as the one of Problem (2.72), should be preferred.

Constrained `EGO` with Support Vector Machines [9]. All the methods that we have seen so far for constrained black-box optimization assume that the measures of the black-box constraints functions $\boldsymbol{g}_\Xi(\boldsymbol{x})$, in $\mathcal{C}_\Xi$ (2.12), are available. Instead, the algorithm in [9] assumes that the only information at our disposal on the black-box constraints is the set $\mathcal{U}_\Xi$ in (2.11). In this setting, we can build a classification problem to predict whether or not a sample $\boldsymbol{x} \in \mathbb{R}^n$ is Ξ-feasible ($u_\Xi(\boldsymbol{x}) = 1$) or not ($u_\Xi(\boldsymbol{x}) = 0$) based on the data in $\mathcal{X}$ (2.9) and $\mathcal{U}_\Xi$ (2.11). In particular, the authors use a modified Probabilistic Support Vector Machine (PSVM) [15, 106] classifier to estimate the probability of Ξ-feasibility[6]:

$$p_N(\boldsymbol{x} \in \Xi) = p\left(\boldsymbol{x} \in \Xi \,\middle|\, \mathcal{U}_\Xi, \mathcal{X}, \boldsymbol{x}\right).$$

The advantages of the classification approach over the Gaussian process regression one (in (2.69)) are [9]:

1. It can easily handle optimization problems with discontinuous or binary black-box constraints;
2. Only a fraction of the black-box constraints might need to be evaluated (if, for any $\boldsymbol{x} \in \mathbb{R}^n, \exists g_\Xi^{(j)}(\boldsymbol{x}) > 0, j \in \{1, \ldots, q_\Xi\}$, then $u_\Xi(\boldsymbol{x}) = 0$), possibly reducing the amount of expensive simulations/experiments;

[6]We use the notation $p\left(\boldsymbol{x} \in \Xi \,\middle|\, \mathcal{U}_\Xi, \mathcal{X}, \boldsymbol{x}\right)$ instead of $p\left(\boldsymbol{x} \in \Xi \,\middle|\, \mathcal{C}_\Xi, \mathcal{X}, \boldsymbol{x}\right)$ (as in (2.69)) to highlight the fact that the estimated probability of Ξ-feasibility is obtained without using the information brought by the measures of the black-box constraints functions in $\mathcal{C}_\Xi$ (2.12), but only using the 0/1 (not Ξ-feasible/Ξ-feasible) information in $\mathcal{U}_\Xi$ (2.11).

3. PSVMs can address dependent constraints without explicit knowledge of their correlation.

At the same time, we point out one major disadvantage: if no Ξ-feasible samples are available after the initial experimental design, then we can only look for one by thoroughly exploring Ω (e.g. by minimizing a suitable exploration function such as (2.56a)), instead of solving Problem (2.71). This can make the initial search for a Ξ-feasible sample particularly inefficient (as we will see in Chapter 6). In any case, the algorithm in [9] assumes that, after the initial experimental design, $\exists \boldsymbol{x}_i \in \mathcal{X}$ in (2.9) such that $\boldsymbol{x}_i \in \Xi$. Then, the new candidate sample $\boldsymbol{x}_{N+1}$ is obtained either by maximizing the constrained expected improvement, defined as:

$$\begin{aligned} a_N(\boldsymbol{x}) &= CEI_N(\boldsymbol{x}) \\ &= p_N(\boldsymbol{x} \in \Xi) \cdot EI_N(\boldsymbol{x}), \end{aligned} \tag{2.73}$$

where, differently from (2.70), the probability of Ξ-feasibility is obtained from the PSVM classifier instead of (2.69), or by solving:

$$\begin{aligned} \boldsymbol{x}_{N+1} = \arg \max_{\boldsymbol{x}} & \; EI_N(\boldsymbol{x}) \\ \text{s.t.} \quad & \boldsymbol{x} \in \Omega \\ & p_N(\boldsymbol{x} \in \Xi) \geq \gamma. \end{aligned} \tag{2.74}$$

In [9], the authors propose to set $\gamma = 0.5$ for Problem (2.74), a common value used when defining the decision boundary of a classifier. The algorithm in [9] also includes a sampling phase aimed towards refining the decision boundary of the PSVM classifier.

Chapter 3. Preference-based optimization

This Chapter delves into Preference-Based Optimization (PBO), a particular branch of optimization tasked with finding the decision vector that is most preferred by a human Decision-Maker (DM). In this context, there is no way to objectively quantify the "degree of goodness" of a certain sample. Instead, the optimization must be carried out by querying the DM, who makes comparisons between different decision vectors and states which of them he/she prefers. As pointed out in [11] (and referenced articles), *it is best to let the individual compare only two samples at a time*, instead of giving him/her multiple options, to avoid the "choice overload" effect and achieve a more reliable response. Given two points $\boldsymbol{x}_i, \boldsymbol{x}_j \in \mathbb{R}^n$, the human decision-maker can either state that $\boldsymbol{x}_i$ is "better than" $\boldsymbol{x}_j$, $\boldsymbol{x}_i$ is "worse than" $\boldsymbol{x}_j$ or that both are "equally as good".

On the surface, it might seem like preference-based optimization is quite different from black-box (or global) optimization. However, we might ask ourselves:

Can an arbitrary criterion of a human-decision maker be "translated" into a mathematical function $f(\boldsymbol{x})$ such that solving the GOP (1.1) *leads to finding his/her most preferred decision vector?*

If that were the case, then some of the results and algorithms that we have seen so far for GO (Chapter 1) and BBO (Chapter 2) could also be adapted for preference-based optimization. From a mathematical point of view, the preferences of the DM can be modeled using the utility theory framework [104]. Most importantly, we will see how, under some assumptions, the (subjective) criterion used by the human decision-maker when comparing the samples can be described by a continuous *utility function* whose maximizer is none other than the decision vector that is most preferred by the DM. Hence, *we can adapt the surrogate-based methods that we have seen in Chapter 2 to the preference-based setting*. This time, instead of approximating the black-box cost function, we need to derive a surrogate model for the utility function of the decision-maker. Furthermore, the latter needs to be estimated using only the preferences expressed by the DM, since there is no way of measuring the "degree of goodness" of a sample.

The surrogate-based optimization scheme described in Algorithm 5 still holds in the preference-based framework, although the concept of sample evaluation is different (see Definition 1.1). Similarly to BBO, performing sample evaluations in the preference-based optimization framework is expensive (cf. Assumption 2.1). For example, suppose that we want to calibrate the controller of some industrial process. It is common in industrial practice to follow a trial-and-error approach: an expert calibrator (DM) tries several different tunings of the regulator by performing a certain number of closed-loop experiments; then, he/she selects the calibration that behaved the best, based on his/her subjective criterion. A better alternative (from an efficiency standpoint) is to employ preference-based optimization to drive the search for the best calibration. An evaluation of a calibration $\boldsymbol{x}_i \in \Omega$ in the preference-based framework is to be interpreted as follows: $\boldsymbol{x}_i$, a tuning that has never been tried, is compared by the DM to a previously seen calibration, $\boldsymbol{x}_j \in \Omega$, to produce a preference (e.g. $\boldsymbol{x}_i$ is "better than" $\boldsymbol{x}_j$). At the same time, the human decision-maker can express his/her judgment on the Ξ-feasibility of $\boldsymbol{x}_i$, e.g. "calibration $\boldsymbol{x}_i$ shows unsatisfactory performances" (i.e. it is Ξ-infeasible). Clearly, the evaluation of $\boldsymbol{x}_i$ might involve one or multiple simulations/experiments. Therefore, we are only interested in those PBO procedures that minimize the number of sample evaluations (such as surrogate-based methods).

Overall, preference-based optimization is more niche than black-box optimization. In the context of control systems, PBO has been successful in controller calibration tasks, see for example [120, 155, 156].

This Chapter is organized as follows. Section 3.1 provides the notions of utility theory that are necessary for the definition of the preference-based optimization problem, which we present in Section 3.2. In particular, we show how the PBO problem is related to the BBO problem. Then, Section 3.3 describes the data resulting from sample evaluations in the preference-based optimization framework. We focus on surrogate-based methods for PBO and follow the same scheme adopted in Chapter 2: in Section 3.4, we show how to adapt the surrogate models reviewed in Section 2.5 to the preference-based setting; then, in Section 3.5, we report the most popular infill sampling criteria. Lastly, Section 3.6 is devoted to summarizing the GO, BBO and PBO frameworks that we have seen so far in this book.

3.1 Notions of (ordinal) utility theory

Utility theory [104] is a framework that models the rationale behind the choices made by individuals. It is widely used in economics to describe consumer behavior. In this book, we will use some notions of (ordinal) utility theory to define the preference-based optimization problem. In PBO, we consider the set Ω of the GOP (1.1) as a set of alternatives among which a decision-maker can choose from. All the tastes of an individual are embedded in a binary relation $\gtrsim$ on Ω (see Appendix A.1), called the (weak) preference relation. Given two alternatives $\boldsymbol{x}_i, \boldsymbol{x}_j \in \Omega$, if $\boldsymbol{x}_i \gtrsim \boldsymbol{x}_j$ holds, then the decision-maker deems $\boldsymbol{x}_i$ "at least as good as" $\boldsymbol{x}_j$. The preference relation $\gtrsim$ is usually "split" into two transitive binary relations:

- The strict preference relation $\succ$ on Ω, i.e. $\boldsymbol{x}_i \succ \boldsymbol{x}_j$ if and only if $\boldsymbol{x}_i \succsim \boldsymbol{x}_j$ but not $\boldsymbol{x}_j \succsim \boldsymbol{x}_i$ ($\boldsymbol{x}_i$ is "better than" $\boldsymbol{x}_j$ or, equivalently, $\boldsymbol{x}_j$ is "worse than" $\boldsymbol{x}_i$), and
- The indifference relation $\sim$ on Ω, i.e. $\boldsymbol{x}_i \sim \boldsymbol{x}_j$ if and only if $\boldsymbol{x}_i \succsim \boldsymbol{x}_j$ and $\boldsymbol{x}_j \succsim \boldsymbol{x}_i$ ($\boldsymbol{x}_i$ is "as good as" $\boldsymbol{x}_j$).

From an economics standpoint, a decision-maker is rational if his/her preference relation satisfies certain properties, as highlighted by the following Definition.

Definition 3.1: Rational decision-maker [104]. *Consider a decision-maker with preference relation $\succsim$ on Ω. We say that the DM is rational (from an economics standpoint) if $\succsim$ is a reflexive, transitive and complete binary relation on Ω.*

We now give some insights on why such properties characterize the rationality of an individual:

- Reflexivity of $\succsim$ on Ω implies that, for the DM, any alternative is as good as itself, i.e. $\boldsymbol{x}_i \succsim \boldsymbol{x}_i$, for each $\boldsymbol{x}_i \in \Omega$;
- A decision-maker whose preference relation $\succsim$ on Ω is transitive is able to express his/her preferences in a coherent manner since, if $\boldsymbol{x}_i \succsim \boldsymbol{x}_j$ and $\boldsymbol{x}_j \succsim \boldsymbol{x}_k$ hold, then $\boldsymbol{x}_i \succsim \boldsymbol{x}_k$, for any $\boldsymbol{x}_i, \boldsymbol{x}_j, \boldsymbol{x}_k \in \Omega$;
- Completeness of $\succsim$ on Ω implies that the DM is able to express a preference between any two alternatives in Ω, i.e. either $\boldsymbol{x}_i \succsim \boldsymbol{x}_j$ or $\boldsymbol{x}_j \succsim \boldsymbol{x}_i$ hold for each $\boldsymbol{x}_i, \boldsymbol{x}_j \in \Omega$.

The transitivity assumption in Definition 3.1 is motivated by the so called *money pump argument*.

Example 3.1: The money pump argument [104]

Let x_1, x_2, x_3 be three different goods. Suppose that a decision-maker with preference relation $\succsim$ ranks the goods in a non-transitive manner as follows:

$$x_1 \succ x_2, \quad x_2 \succsim x_3, \quad x_3 \sim x_1.$$

The transitivity assumption is violated by the third preference, which should be $x_1 \succ x_3$ to be coherent with the former two. Now, suppose that the decision-maker owns x_1 and is willing to exchange a good for any other that is weakly preferred. Moreover, the DM is willing to trade a good and pay a fee for another commodity that is strictly preferred. Then, due to the non-transitivity of $\succsim$, the decision-maker can be exploited into forfeiting an arbitrary amount of money. For example:

1. The DM trades x_1 with x_3 for free (indifference);

2. Then, x_3 is exchanged with x_2 for free (weak preference);

3. Finally, the decision-maker trades x_2 with x_1 and pays a fee (strict preference).

4. The same process can be repeated from Step 1, until the DM runs out of money.

Clearly, this is not a rational behavior.

At the same time, we could make an argument against transitivity for rational decision-makers as follows.

Example 3.2: Just noticeable differences and transitivity [39]

Let x_1 be a cup of coffee with a grain of sugar in it, x_2 be a cup of coffee with two grains of sugar in it, and so on for any $x_k, k \in \mathbb{N}$. Surely, a decision-maker cannot taste the difference between x_k and x_{k+1} for any $k \in \mathbb{N}$, therefore $x_k \sim x_{k+1}$. Then, due to the transitive property of $\sim$, we have:

$$x_1 \sim x_2 \sim \ldots \sim x_{1000000} \sim \ldots$$

which is clearly not true for a rational decision-maker.
Problems of this type arise from the transitivity of the indifference relation $\sim$ and are related to the concept of just-noticeable differences. In particular, consider any experiment that involves a sensory evaluation performed by an individual (such as tasting a cup of coffee) and that depends on some parameter (e.g. the grains of sugar). Then, the individual requires a minimum level of stimulation (or, rather, a minimum rate of change on the parameter) to discern between any two experiments (e.g. it might take at least a hundred grains of sugar to taste the difference between two cups of coffee). See [19] for a more detailed example.

Among the three properties required by Definition 3.1, completeness is the strongest assumption and might not necessarily describe a decision-maker that is rational ("in a moral sense").

Example 3.3: Completeness and rationality [39]

A man is given the choice between shooting his dog or shooting his cat. Clearly, even if the decision-maker is rational, he might not be able to choose.
Incomplete preference relations often arise when making decisions over highly hypothetical situations (which might never happen in real life) or when the DM is asked to make very complex choices, too complex for "intuitive insight" [7].

Even though there are a few caveats on the properties required by Definition 3.1, *for the remainder of* *this book we will assume that the decision-maker who expresses the preferences in PBO is rational* in that regard.

Reflexivity, transitivity and completeness of $\gtrsim$ on Ω play a key role when describing the preference relation through a so-called utility function. An utility function is a function $u_{\gtrsim} : \Omega \to \mathbb{R}$ that assigns to an alternative $\boldsymbol{x}_i$ a (strictly) higher value than that of an alternative $\boldsymbol{x}_j$ if and only if $\boldsymbol{x}_i$ is ranked (strictly) above $\boldsymbol{x}_j$ by the preference relation. More formally:

Definition 3.2: Representability of $\gtrsim$ and utility function [104]. *Let Ω be a nonempty set and $\gtrsim$ be a preference relation on Ω. We say that $u_{\gtrsim} : \Omega \to \mathbb{R}$ represents $\gtrsim$ on Ω if $u_{\gtrsim}(\cdot)$ is an order-preserving function, that is if:*

$$\boldsymbol{x}_i \gtrsim \boldsymbol{x}_j \quad \text{if and only if} \quad u_{\gtrsim}(\boldsymbol{x}_i) \geq u_{\gtrsim}(\boldsymbol{x}_j), \tag{3.1a}$$

$$\boldsymbol{x}_i \succ \boldsymbol{x}_j \quad \text{if and only if} \quad u_{\gtrsim}(\boldsymbol{x}_i) > u_{\gtrsim}(\boldsymbol{x}_j), \tag{3.1b}$$

$$\boldsymbol{x}_i \sim \boldsymbol{x}_j \quad \text{if and only if} \quad u_{\gtrsim}(\boldsymbol{x}_i) = u_{\gtrsim}(\boldsymbol{x}_j), \tag{3.1c}$$

for any $\boldsymbol{x}_i, \boldsymbol{x}_j \in \Omega$. If such a function exists, then $\gtrsim$ is said to be representable, and $u_{\gtrsim}(\cdot)$ is called an utility function for $\gtrsim$ on Ω.

Remark 3.1 (Uniqueness of the utility function [104])**.** *Let $\gtrsim$ be a representable preference relation on $\Omega \neq \emptyset$ and $u_{\gtrsim}(\cdot)$ be its corresponding utility function. Then, $u_{\gtrsim}(\cdot)$ is unique up to any strictly increasing transformation.*

There are a lot of results in the literature concerning the representability of the preference relation, which differ on the assumptions made on Ω, see for example [31, 40, 104]. Some theorems assume Ω to be either finite, countable or, in general, any metric space. For PBO, we restrict ourselves to the case $\Omega \subset \mathbb{R}^n$. Before actually stating the representability theorem of interest, we introduce the notion of continuity of preference relations in metric spaces (such as Ω).

Definition 3.3: $\gtrsim$-contour sets [104]. *Let $\gtrsim$ be a preference relation on Ω. For any $\boldsymbol{x} \in \Omega$, the weak and strict upper $\gtrsim$-contour sets of $\boldsymbol{x}$ are defined as:*

$$\mathcal{U}_{\gtrsim}(\boldsymbol{x}) = \{\tilde{\boldsymbol{x}} : \tilde{\boldsymbol{x}} \in \Omega, \tilde{\boldsymbol{x}} \gtrsim \boldsymbol{x}\}, \tag{3.2a}$$

$$\mathcal{U}_{\succ}(\boldsymbol{x}) = \{\tilde{\boldsymbol{x}} : \tilde{\boldsymbol{x}} \in \Omega, \tilde{\boldsymbol{x}} \succ \boldsymbol{x}\}, \tag{3.2b}$$

respectively. The weak and strict lower $\gtrsim$-contour sets of $\boldsymbol{x}$ are defined analogously:

$$\mathcal{L}_{\gtrsim}(\boldsymbol{x}) = \{\tilde{\boldsymbol{x}} : \tilde{\boldsymbol{x}} \in \Omega, \boldsymbol{x} \gtrsim \tilde{\boldsymbol{x}}\}, \tag{3.3a}$$

$$\mathcal{L}_{\succ}(\boldsymbol{x}) = \{\tilde{\boldsymbol{x}} : \tilde{\boldsymbol{x}} \in \Omega, \boldsymbol{x} \succ \tilde{\boldsymbol{x}}\}. \tag{3.3b}$$

Definition 3.4: Continuity of a preference relation [104]. *Let Ω be a metric space and $\succsim$ be a preference relation on Ω. We say that:*

- *$\succsim$ is upper semi-continuous if $\mathcal{L}_{\succ}(\boldsymbol{x})$ is an open subset of Ω for each $\boldsymbol{x} \in \Omega$,*
- *$\succsim$ is lower semi-continuous if $\mathcal{U}_{\succ}(\boldsymbol{x})$ is an open subset of Ω for each $\boldsymbol{x} \in \Omega$,*
- *$\succsim$ is continuous if it is both upper and lower semi-continuous.*

Intuitively speaking, if $\succsim$ is an upper semi-continuous preference relation on a metric space Ω and if $\boldsymbol{x}_i \succ \boldsymbol{x}_j$, then an alternative $\boldsymbol{x}_k$ which is "very close" to $\boldsymbol{x}_j$ should also be deemed strictly worse than $\boldsymbol{x}_i$, i.e. $\boldsymbol{x}_i \succ \boldsymbol{x}_k$. Similarly for the other continuity definitions. In some sense, there are "no jumps" between the preferences expressed by the DM.
We are now ready to state Debreu's utility representation Theorem for $\mathbb{R}^n$.

Theorem 3.1: Debreu's utility representation Theorem for $\mathbb{R}^n$ [31]

Let Ω be any nonempty subset of $\mathbb{R}^n$ and $\succsim$ be a preference relation on Ω of a rational decision-maker. There exists a continuous (respectively, upper semi-continuous) utility representation for $\succsim$ if and only if $\succsim$ is continuous (upper semi-continuous).

Theorem 3.1 is quite powerful; basically, it states that the (subjective) criterion used by any rational decision-maker (as in Definition 3.1) when making choices among alternatives can be described using a continuous mathematical function $u_{\succsim}(\boldsymbol{x})$.
As a final remark, note that there also exist representations for preference relations that are not complete and/or transitive, see for example [99, 103].

3.2 The preference-based optimization problem

In this Section, we use the notions of utility theory that we have just reviewed to define the preference-based optimization problem. Recall that the goal of PBO is to find the decision vector that is most preferred by a human decision-maker. Let $\succsim$ be the preference relation on Ω associated to the DM. Similarly to BBO methods, most preference-based optimization algorithms assume that Ω is either defined as in (1.3) or in (1.5) (no equality constraints). We can state the preference-based optimization problem as follows:

$$\text{find } \boldsymbol{x}^* \in \Omega \text{ such that } \boldsymbol{x}^* \succsim \boldsymbol{x}, \forall \boldsymbol{x} \in \Omega. \tag{3.4}$$

$\boldsymbol{x}^*$ in (3.4) is referred to as a $\succsim$-maximum of Ω (see Appendix 3.1 for more thorough definitions of maxima, minima, suprema and infima of Ω ordered by $\succsim$). In practice, there could be more than one

$\gtrsim$-maximum of Ω[1]. Therefore, we define the set of all the alternatives that are "optimal" from the decision-maker's point of view as:

$$\mathcal{X}^* = \left\{\boldsymbol{x}_i^* : \boldsymbol{x}_i^* \in \Omega, \nexists \boldsymbol{x} \in \Omega \text{ s.t. } \boldsymbol{x} \succ \boldsymbol{x}_i^*\right\}, \tag{3.5}$$

which plays the same role as the set of global minimizers of the GOP (1.1) in (1.2). In economics, the problem of finding $\mathcal{X}^*$ in (3.5) is often referred to as a choice problem [104]. Concerning the existence of a $\gtrsim$-maximum of Ω, we can state the following Proposition, which can be seen as a generalization of the Extreme Value Theorem 1.1 for preference relations.

Proposition 3.1: Existence of a $\gtrsim$-maximum of Ω [104]. *A $\gtrsim$-maximum of Ω is guaranteed to exist if Ω is a compact subset of a metric space (in our case $\Omega \subset \mathbb{R}^n$) and $\gtrsim$ is a continuous preference relation on Ω of a rational decision-maker (as in Definition 3.1).*

Instead of maximizing a binary relation $\gtrsim$ on Ω, we would rather build a more "traditional" optimization problem in such a way that its solutions are those decision vectors that are the most preferred by the DM, see (3.5). To do so, note that the conditions which guarantee the existence of a $\gtrsim$-maximum of Ω in Proposition 3.1 also make the preference relation representable by Theorem 3.1. Therefore, by Definition 3.2, Problem (3.4) is equivalent to maximizing the utility function of the decision-maker over Ω:

$$\begin{aligned} \mathcal{X}^* = \arg &\max_{\boldsymbol{x}} u_{\gtrsim}(\boldsymbol{x}) \\ \text{s.t.} &\quad \boldsymbol{x} \in \Omega. \end{aligned} \tag{3.6}$$

In practice, the latter problem is equivalent to the global optimization problem proposed in Section 1.1, as pointed out by the following Remark.

Remark 3.2 (Relationship to the global optimization problem and scoring function)**.** *Problem* (3.6) *can be re-written as the GOP* (1.1) *by taking $f(\boldsymbol{x}) = -u_{\gtrsim}(\boldsymbol{x})$. We refer to $f(\boldsymbol{x})$ in the PBO framework as the scoring function (of the decision-maker).*
Formally, $f(\boldsymbol{x})$ in the GOP (1.1) *and $u_{\gtrsim}(\boldsymbol{x})$ obtained by Definition 3.2 have different domains ($\mathbb{R}^n$ and Ω respectively). However, assuming that $u_{\gtrsim}(\boldsymbol{x})$ is continuous and Ω is a compact subset of $\mathbb{R}^n$ (which are either results or assumptions of Proposition 3.1 and Theorem 3.1), by Tietze Extension Theorem (Theorem A.6), there exists a continuous extension of $u_{\gtrsim}(\boldsymbol{x})$ with domain $\mathbb{R}^n$.*

[1]A slight clarification: consider a partially ordered set (poset) $(\mathcal{A}, \mathcal{R})$, where $\mathcal{R}$ is a generic binary relation (formally, a partial order) on $\mathcal{A}$, see Definition A.4. In Appendix A.1, we have reported that any nonempty subset of a poset can have at most one $\mathcal{R}$-maximum (see Definition A.7). However, $(\Omega, \gtrsim)$ is not necessarily a partially ordered set since $\gtrsim$ on Ω need not be antisymmetric. Hence, if $(\Omega, \gtrsim)$ is a preordered set instead of a poset, there could be multiple $\gtrsim$-maxima of Ω.

Now that we have defined the preference-based optimization problem in detail, we highlight the assumptions made by PBO methods when solving the GOP (1.1).

Assumption 3.1 (Assumptions for preference-based optimization)**.** *Consider the global optimization problem in* (1.1) *with* Ω *defined as in* (1.5)*. In the context of preference-based optimization, the cost function and (possibly) some of the constraints functions of the GOP* (1.1) *are assumed to be black-boxes (see Assumption 2.1). In particular, in the PBO setting, the cost function is actually the scoring function (see Remark 3.2) of a human decision-maker with preference relation* $\gtrsim$ *on* Ω*, for which no analytical formulation is available. Furthermore, there is no way to objectively quantify* $f(\boldsymbol{x})$*. Instead, the optimization must be carried out by asking the DM to compare couples of different samples. Given two samples* $\boldsymbol{x}_i, \boldsymbol{x}_j \in \Omega$*, the only information that we can extract on the cost function in the PBO framework is which among the following statements holds:* $\boldsymbol{x}_i \succ \boldsymbol{x}_j$*,* $\boldsymbol{x}_i \sim \boldsymbol{x}_j$ *or* $\boldsymbol{x}_j \succ \boldsymbol{x}_i$*.*

The previous Assumption shows how *preference-based optimization can be interpreted as a particular instance of black-box optimization where the cost function is both a black-box and cannot be quantified objectively* (cf. Assumption 2.1). Furthermore, Assumption 3.1 points out that a PBO problem could also include some black-box constraints which, consistently with Chapter 2, are described by the set Ξ in (2.2). Hence, the preference-based optimization problem is none other than:

$$\begin{aligned} \mathcal{X}^* &= \arg\min_{\boldsymbol{x}} f(\boldsymbol{x}) \\ \text{s.t.} & \quad \boldsymbol{x} \in \Omega \cap \Xi, \end{aligned}$$

which is mathematically equivalent to the black-box optimization problem in (2.1), but this time $f(\boldsymbol{x})$ is the scoring function of the DM and not a measurable black-box function.
Similarly to BBO, we differentiate between unconstrained and constrained preference-based optimization problems as follows (cf. Definition 2.1).

Definition 3.5: Unconstrained and constrained BBO. *Regardless of the structure of* Ω *for the GOP* (2.1)*, we say that a method belongs to the unconstrained preference-based optimization framework if it assumes that* $\Xi = \mathbb{R}^n$*. Vice-versa, whenever* Ξ *is defined as in* (2.2)*, we speak of constrained preference-based optimization.*

In the context of constrained PBO, we also highlight a particular type of black-box constraints, which we refer to as decision-maker-based constraints.

Definition 3.6: Decision-maker-based constraint. *A decision-maker-based constraint* $\tilde{g}_{\Xi}(\boldsymbol{x}) \leq 0, \tilde{g}_{\Xi} : \mathbb{R}^n \to \{0, 1\}$*, is a binary, relaxable, black-box and known constraint (according to the* `QRAK` *taxonomy in Definition 2.3) which is completely defined by a human decision-maker. A*

DM-based constraint works as follows: given a sample $\boldsymbol{x}_i \in \mathbb{R}^n$, the decision-maker states whether it is feasible ($\tilde{g}_\Xi(\boldsymbol{x}_i) = 0$) or not ($\tilde{g}_\Xi(\boldsymbol{x}_i) = 1$), depending on his/her subjective criterion.

A decision-maker-based constraint can be seen as asking the DM a "yes/no question", which is different from expressing a preference between two alternatives. For example, decision-maker-based constraints have been included in [156] when tuning the Model Predictive Controller [110] for the lane-keeping and obstacle-avoidance tasks in autonomous driving vehicles. In particular, after performing a simulation using a certain decision vector, the calibrator assesses whether or not the tuning is acceptable and if the controller achieves satisfactory performances. Acceptability and satisfiability can be seen as two separate DM-based constraints.

We conclude this Section with an Example of a preference-based optimization problem in the context of control systems. In the next Example, we also give some insights on the relationship between preference-based optimization and multi-objective optimization.

Example 3.4: Preference-based optimization for control systems

We go back to the control systems example presented in Chapter 2 (i.e. Example 2.1). We have seen how, by defining suitable performance indicators, we can employ a black-box optimization procedure to calibrate the controller's parameters, effectively driving the closed-loop experiments on the control system.

In many cases, only qualitative control specifications are available, making the choice of $f(\boldsymbol{x})$ and Ξ for the GOP (2.1) less straightforward. Instead of going through the trouble of choosing the most appropriate performance indicators, we rely on an expert calibrator, who knows the system well and has a clear objective in mind. We use a preference-based optimization procedure to help him/her with the tuning process. Consistently with Example 2.1, suppose that the calibrator assesses the performances of a certain parametrization of the controller through a step test. For instance, he/she could take a look at the output signal and control action resulting from a calibration $\boldsymbol{x}_i$ and compare it to the results of a previously tested tuning $\boldsymbol{x}_j$, as depicted in Figure 8. Clearly, if the decision-maker expresses his/her feedback purely by looking at the curves in Figure 8, then there is no need to repeat the same experiment twice, we just have to save the $output(t; \boldsymbol{x}_i)$ and $control\_action(t; \boldsymbol{x}_i)$ signals when the calibration $\boldsymbol{x}_i$ is tested.

In some sense, the criterion used by the DM could be (implicitly) a trade-off between the performance indicators presented in Example 2.1, such as:

$$f(\boldsymbol{x}) = \omega_{t_{rise}} \cdot t_{rise}(\boldsymbol{x}) + \omega_{tsv} \cdot tsv(\boldsymbol{x}), \tag{3.7}$$

where $\omega_{t_{rise}}, \omega_{tsv} \in \mathbb{R}_{>0}$ are two positive weights for the settling time and the total square variation respectively. Clearly, $\omega_{t_{rise}}$ and ω_{tsv} are unknown and practically depend on the DM who expresses the preferences. Going back to Figure 8, a calibrator who prioritizes the rise time ($\omega_{t_{rise}} \gg \omega_{tsv}$) would say that $\boldsymbol{x}_i \succ \boldsymbol{x}_j$. Vice-versa, a decision-maker that prefers moderate control actions ($\omega_{tsv} \gg \omega_{t_{rise}}$) deems $\boldsymbol{x}_j \succ \boldsymbol{x}_i$.

The derivation of the cost function $f(\boldsymbol{x})$ in (3.7) is closely related to *Multi-Objective Optimization (MOO)* [84]. As a matter of fact, it is a way of scalarizing a multi-objective optimization problem. *Decision-makers also play a key-role in MOO*. For example, they choose which solution, among a set of Pareto optimal ones, is the most suited for the task at hand. Alternatively, DMs can specify the weights $\omega_{t_{rise}}, \omega_{tsv}$ for $f(\boldsymbol{x})$ in (3.7). *Although both preference-based optimization and multi-objective optimization rely on decision-makers, we stress that, differently from MOO, in PBO there is no explicit cost function to be minimized.*

Lastly, we point out that a possible decision-maker-based constraint could be something like: "the output signals must not show pronounced oscillations". If that were the case, then calibration $\boldsymbol{x}_i$ in Figure 8 would be deemed as Ξ-infeasible while $u_{\Xi}(\boldsymbol{x}_j) = 1$.

3.3 Data available for preference-based optimization

In the previous Section, we have seen how preference-based optimization shares many similarities with black-box optimization, the only difference being the information available on the cost function $f(\boldsymbol{x})$ of the GOP (2.1). In particular, the set of cost function measures $\mathcal{Y}$ in (2.10) is not available. Instead, we must rely on the preferences expressed by the human decision-maker which, for the sake of simplicity, are described by a so-called preference function defined as follows.

Definition 3.7: Preference function [11]. *Let $\gtrsim$ be a preference relation on Ω of a human decision-maker and $f(\boldsymbol{x})$ be its corresponding scoring function. We define the* <u>*preference*</u>

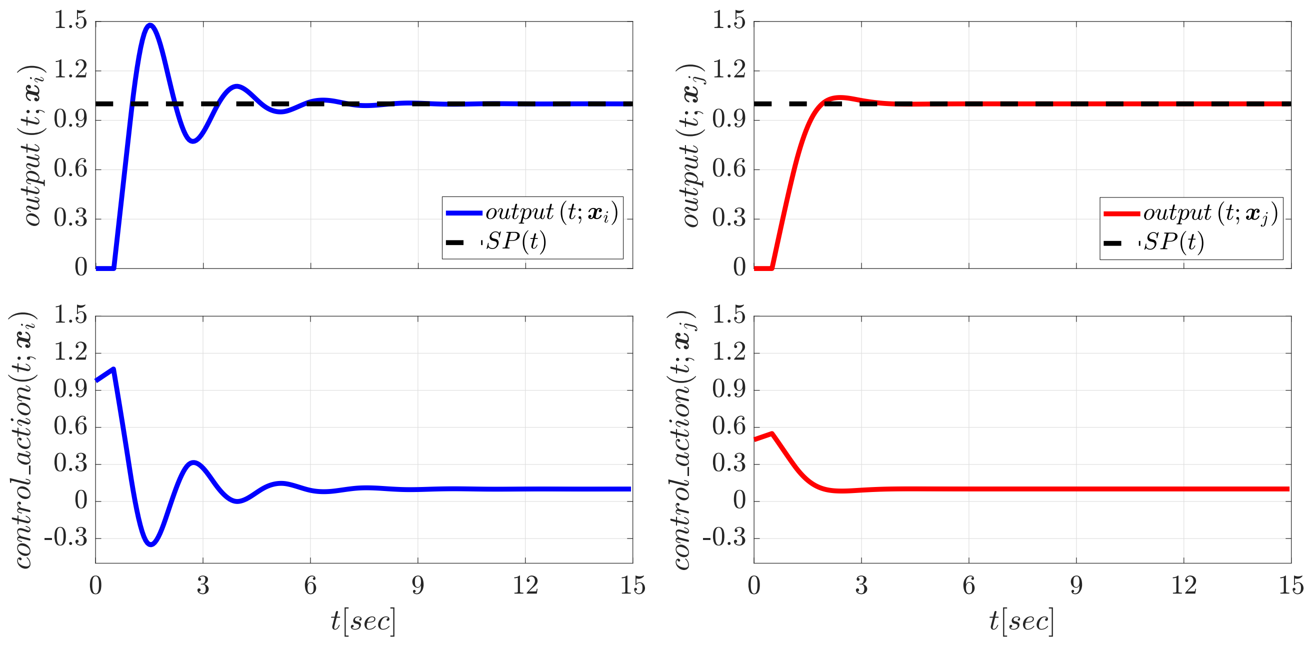


Figure 8: Query window for preference-based optimization. The decision-maker is presented with the output and control action signals achieved by two different calibrations, x_i (left) and x_j (right).

function $\pi_{\gtrsim} : \mathbb{R}^n \times \mathbb{R}^n \to \{-1, 0, 1\}$ *as*[a]*:*

$$\pi_{\gtrsim}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \begin{cases} -1 & \text{if } \boldsymbol{x}_i \succ \boldsymbol{x}_j \Leftrightarrow f(\boldsymbol{x}_i) < f(\boldsymbol{x}_j) \\ 0 & \text{if } \boldsymbol{x}_i \sim \boldsymbol{x}_j \Leftrightarrow f(\boldsymbol{x}_i) = f(\boldsymbol{x}_j) \\ 1 & \text{if } \boldsymbol{x}_j \succ \boldsymbol{x}_i \Leftrightarrow f(\boldsymbol{x}_i) > f(\boldsymbol{x}_j) \end{cases} . \tag{3.8}$$

[a]We stick to the definition of the preference function in [11], where it is derived without any notion of utility theory. Therein, the authors simply assume that there exists a scoring function $f : \mathbb{R}^n \to \mathbb{R}$ that ranks the alternatives in Ω as a decision-maker with preference relation $\gtrsim$ on Ω would. Formally, to be precise, we should assume Ω to be a closed subset of $\mathbb{R}^n$ (due to Tietze Extension Theorem A.6), the DM to be rational and $\gtrsim$ on Ω to be continuous in order to ensure the existence of $f(\boldsymbol{x})$ in (3.8) (see Section 3.1 and Section 3.2).

In preference-based optimization, the data resulting from N sample evaluations is composed of:

1. The set of samples $\mathcal{X}$ in (2.9),

2. The preferences expressed by the human decision-maker. In particular, we use two sets to describe them:

$$\mathcal{B} = \{b_h : h = 1, \ldots, M, b_h \in \{-1, 0, 1\}\}, \tag{3.9}$$

where b_h is the h-th preference obtained by comparing a certain couple of samples, and:

$$\mathcal{S} = \Big\{ (\ell(h), \kappa(h)) : h = 1, \ldots, M, \ell(h), \kappa(h) \in \mathbb{N},$$
$$b_h = \pi_{\gtrsim}\left(\boldsymbol{x}_{\ell(h)}, \boldsymbol{x}_{\kappa(h)}\right), \tag{3.10}$$

$$b_h \in \mathcal{B}, \boldsymbol{x}_{\ell(h)}, \boldsymbol{x}_{\kappa(h)} \in \mathcal{X}\Big\}.$$

$\mathcal{S}$ in (3.10) is a mapping set that highlights which samples in $\mathcal{X}$ (2.9) have been compared to obtain the preferences in $\mathcal{B}$ (3.9). In particular, $\ell : \mathbb{N} \to \mathbb{N}$ and $\kappa : \mathbb{N} \to \mathbb{N}$ are two mapping functions that associate the indexes of the samples in $\mathcal{X}$ to the preferences in $\mathcal{B}$. Note that a total of $M \in \mathbb{N}$ preferences have been expressed on the N samples in $\mathcal{X}$.

3. The information on the black-box constraints in Ξ, i.e. either the set $\mathcal{U}_\Xi$ in (2.11), which only highlights whether the samples in $\mathcal{X}$ are Ξ-feasible or not, or the set of measures $\mathcal{C}_\Xi$ in (2.12).

The cardinalities of the aforementioned sets are $|\mathcal{X}| = |\mathcal{C}_\Xi| = |\mathcal{U}_\Xi| = N$ and $|\mathcal{B}| = |\mathcal{S}| = M$, with $1 \leq M \leq \begin{pmatrix} N \\ 2 \end{pmatrix}$.

The data that we have just covered in this Section is used by surrogate-based methods to solve the preference-based optimization problem, i.e. the GOP (2.1). PBO procedures follow the same scheme reported in Algorithm 5 but, in this case, they rely on a surrogate model for the scoring function of the decision-maker, which is estimated from the preferences reported in sets $\mathcal{B}$ (3.9) and $\mathcal{S}$ (3.10).

3.4 Surrogate models

The problem of estimating predictive models from the preferences expressed by a human decision-maker, namely preference learning, has received much attention in the machine learning community. In [46], the authors distinguish between *learning preference relations* and *learning utility functions*, pointing out their applications to different machine learning tasks. Preference learning has also been applied in the field of reinforcement learning to take into account qualitative feedback provided by experts (decision-makers) [13, 47]. Going back to the preference-based optimization framework, most procedures use a predictive (surrogate) model for the latent utility function of the DM although, differently from preference learning, its prediction accuracy is not the main concern. In practice, *we only need a surrogate model that is expressive enough to locate the global minimizers of the GOP* (2.1) *and not necessarily accurate on the whole constraint set* Ω [11, 21]. At the moment, compared to black-box optimization, very few preference-based optimization algorithms exist. Most procedures use a surrogate model that relies on Gaussian processes [12, 21, 52]. Conversely, in [11], the authors propose an approximation of the latent scoring function that is based on radial basis functions.

Remark 3.3. *In what follows, consistently with Section 2.5, we denote the surrogate models for the scoring function as $\hat{f}_N(\boldsymbol{x})$, possibly highlighting their hyper-parameters. Formally, a more proper notation in the preference-based setting would be $\hat{f}_M(\boldsymbol{x})$, since N sample evaluations produce M*

preferences (see $\mathcal{B}$ in (3.9) *and $\mathcal{S}$ in* (3.10)*), from which the surrogate models are estimated. However, as we will see later on in this book, most PBO algorithms are such that $M = N - 1$. That is because, often, the decision-maker is asked to compare the current best candidate $\boldsymbol{x_{best}}(N)$ with one, and only one, other sample $\boldsymbol{x}_{N+1} \neq \boldsymbol{x_{best}}(N)$. Hence, at each iteration of Algorithm 5, we evaluate only one sample and observe only one preference.*
On a side note, we also point out that the notation $\hat{f}_N(\boldsymbol{x})$ better captures the fact that the surrogate models are linear combinations of N basis functions, as we will see shortly.

3.4.1 Surrogate scoring function based on RBFs

Consider the radial basis function expansion surrogate model in (2.20). When dealing with the problem of learning the latent scoring function of a human decision-maker, we cannot compute the vector of weights $\boldsymbol{\beta}_f$ for $\hat{f}_N\left(\boldsymbol{x}; \boldsymbol{\beta}_f, \epsilon_f\right)$ in (2.20) by enforcing the interpolation conditions in (2.25), since the measures of the cost function are not available. Instead, in the context of PBO, *we are interested in a surrogate model that correctly describes the preferences expressed by the DM*. We consider the surrogate preference function $\hat{\pi}_{\succsim_N} : \mathbb{R}^n \times \mathbb{R}^n \to \{-1, 0, 1\}$ defined in [11]:

$$\hat{\pi}_{\succsim_N}\left(\boldsymbol{x}_i, \boldsymbol{x}_j; \boldsymbol{\beta}_f, \epsilon_f\right) = \begin{cases} -1 & \text{if } \hat{f}_N\left(\boldsymbol{x}_i; \boldsymbol{\beta}_f, \epsilon_f\right) - \hat{f}_N\left(\boldsymbol{x}_j; \boldsymbol{\beta}_f, \epsilon_f\right) \leq -\sigma_\pi \\ 0 & \text{if } \left|\hat{f}_N\left(\boldsymbol{x}_i; \boldsymbol{\beta}_f, \epsilon_f\right) - \hat{f}_N\left(\boldsymbol{x}_j; \boldsymbol{\beta}_f, \epsilon_f\right)\right| \leq \sigma_\pi \\ 1 & \text{if } \hat{f}_N\left(\boldsymbol{x}_i; \boldsymbol{\beta}_f, \epsilon_f\right) - \hat{f}_N\left(\boldsymbol{x}_j; \boldsymbol{\beta}_f, \epsilon_f\right) \geq \sigma_\pi \end{cases} . \tag{3.11}$$

Differently from $\pi_{\succsim}(\boldsymbol{x}_i, \boldsymbol{x}_j)$ in (3.8), $\hat{\pi}_{\succsim_N}(\boldsymbol{x}_i, \boldsymbol{x}_j; \boldsymbol{\beta}_f, \epsilon_f)$ in (3.11) uses a tolerance $\sigma_\pi \in \mathbb{R}_{>0}$ to avoid strict inequalities and equalities and replaces $f(\boldsymbol{x})$ with the surrogate model for the scoring function. $\hat{\pi}_{\succsim_N}\left(\boldsymbol{x}_i, \boldsymbol{x}_j; \boldsymbol{\beta}_f, \epsilon_f\right)$ in (3.11) correctly describes the preferences in $\mathcal{B}$ (3.9) and $\mathcal{S}$ (3.10) if:

$$b_h = \hat{\pi}_{\succsim_N}\left(\boldsymbol{x}_{\ell(h)}, \boldsymbol{x}_{\kappa(h)}; \boldsymbol{\beta}_f, \epsilon_f\right), \quad \forall b_h \in \mathcal{B}, (\ell(h), \kappa(h)) \in \mathcal{S}, h = 1, \dots, M.$$

This, in turn, translates into some constraints on the surrogate model $\hat{f}_N\left(\boldsymbol{x}; \boldsymbol{\beta}_f, \epsilon_f\right)$ in (2.20), which can be used to find $\boldsymbol{\beta}_f$. In order to do so, in [11], the authors define the following convex optimization problem:

$$\begin{aligned} &\arg\min_{\boldsymbol{\varepsilon}_f, \boldsymbol{\beta}_f} \frac{\lambda_f}{2} \cdot \boldsymbol{\beta}_f^\top \cdot \boldsymbol{\beta}_f + \boldsymbol{r}_f^\top \cdot \boldsymbol{\varepsilon}_f && (3.12) \\ \text{s.t.} \quad &\hat{f}_N\left(\boldsymbol{x}_{\ell(h)}; \boldsymbol{\beta}_f, \epsilon_f\right) - \hat{f}_N\left(\boldsymbol{x}_{\kappa(h)}; \boldsymbol{\beta}_f, \epsilon_f\right) \leq -\sigma_\pi + \varepsilon_f^{(h)} && \forall h : b_h = -1 \\ &\left|\hat{f}_N\left(\boldsymbol{x}_{\ell(h)}; \boldsymbol{\beta}_f, \epsilon_f\right) - \hat{f}_N\left(\boldsymbol{x}_{\kappa(h)}; \boldsymbol{\beta}_f, \epsilon_f\right)\right| \leq \sigma_\pi + \varepsilon_f^{(h)} && \forall h : b_h = 0 \\ &\hat{f}_N\left(\boldsymbol{x}_{\ell(h)}; \boldsymbol{\beta}_f, \epsilon_f\right) - \hat{f}_N\left(\boldsymbol{x}_{\kappa(h)}; \boldsymbol{\beta}_f, \epsilon_f\right) \geq \sigma_\pi - \varepsilon_f^{(h)} && \forall h : b_h = 1 \end{aligned}$$

$$\varepsilon_f^{(h)} \geq 0$$
$$h = 1, \dots, M,$$

where:

- $\boldsymbol{\varepsilon}_f = \begin{bmatrix} \varepsilon_f^{(1)} & \dots & \varepsilon_f^{(M)} \end{bmatrix}^\top \in \mathbb{R}^M_{\geq 0}$ is a vector of slack variables (one for each preference) which takes into consideration that: (i) there might be some outliers in $\mathcal{B}$ (3.9) and $\mathcal{S}$ (3.10) if the human decision-maker expresses the preferences in an inconsistent way, and (ii) the selected radial function and/or shape parameter for $\hat{f}_N\left(\boldsymbol{x}; \boldsymbol{\beta}_f, \epsilon_f\right)$ in (2.20) do not allow a proper approximation of the scoring function $f(\boldsymbol{x})$;

- $\boldsymbol{r}_f = \begin{bmatrix} r_f^{(1)} & \dots & r_f^{(M)} \end{bmatrix}^\top \in \mathbb{R}^M_{>0}$ is a vector of weights which can be used to penalize more some slacks related to the most important preferences;

- $\lambda_f \in \mathbb{R}_{\geq 0}$ plays the role of a regularization parameter. It is easy to see that, for $\lambda_f = 0$, Problem (3.12) is a Linear Program (LP) while, for $\lambda_f > 0$, it is a Quadratic Program (QP).

Note that Problem (3.12) ensures that, at least approximately, $\hat{f}_N\left(\boldsymbol{x}; \boldsymbol{\beta}_f, \epsilon_f\right)$ in (2.20) is a suitable representation of the unknown preference relation $\gtrsim$ on Ω which generated the data (see Theorem 3.1). As a final remark, as we have seen in Section 2.5.1, the quality of the surrogate model depends on the choice of the shape parameter ϵ_f. In [11], the authors recalibrate ϵ_f through grid search leave-one-out cross-validation. We will cover Problem (3.12) and the recalibration of the shape parameter in greater detail in Chapter 4.

3.4.2 Surrogate scoring function based on $\mathcal{GP}$s

Gaussian processes have been widely used in the context of preference learning. The most popular $\mathcal{GP}$-based surrogate model for the scoring function of a human decision-maker is the one proposed in [25]. Therein, the authors consider only the strict preference relation $\succ$ instead of $\gtrsim$ (the indifference relation $\sim$ is not handled explicitly). Under this assumption, it is possible to define the mapping functions $\ell(h), \kappa(h)$ in (3.10) so that:

$$\boldsymbol{x}_{\ell(h)} \succ \boldsymbol{x}_{\kappa(h)}, \quad \forall h = 1, \dots, M,$$

making the set $\mathcal{B}$ in (3.9) redundant.

Similarly to what we have seen in Section 2.5.2, we impose a $\mathcal{GP}$ *prior* on the scoring function $f(\boldsymbol{x})$ of the GOP (2.1) as in (2.34). Hence, the probability distribution of $\boldsymbol{f}$ in (2.36) is the same as the one in (2.38), namely:

$$p\left(\boldsymbol{f};\boldsymbol{\theta}_f\right) = \mathcal{N}\left(\boldsymbol{\mu}_f, \Sigma_f\right), \tag{3.13a}$$

$$\boldsymbol{\mu}_f = \mathbf{0}_N, \tag{3.13b}$$

$$\Sigma_f = K_f\left(\boldsymbol{\theta}_f\right). \tag{3.13c}$$

Coherently with Assumption 2.5, the scoring function $f(\boldsymbol{x})$ is assumed to be affected by zero-mean Gaussian noise noise such that:

$$\begin{aligned} y_{\ell(h)} &= f\left(\boldsymbol{x}_{\ell(h)}\right) + \eta_{f_{\ell(h)}} \\ &= f_{\ell(h)} + \eta_{f_{\ell(h)}}, \quad \eta_{f_{\ell(h)}} \overset{i.i.d.}{\sim} \mathcal{N}\left(0, \sigma_f^2\right), \forall h = 1, \ldots, M, \end{aligned}$$

and similarly for those measures indexed by $\kappa(h)$. However, in the preference-based setting, the measures $y_{\ell(h)}, y_{\kappa(h)}, h = 1, \ldots, M$, are not available. Instead, we only observe the preferences expressed by the decision-maker which, consistently with $\pi_{\gtrsim}\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right)$ in (3.8), we assume to be such that:

$$\boldsymbol{x}_{\ell(h)} \succ \boldsymbol{x}_{\kappa(h)} \text{ if and only if } y_{\ell(h)} < y_{\kappa(h)}.$$

The addition of the noise serves to capture possible inconsistencies in the preferences expressed by the DM (similarly to the role of the slacks in Problem (3.12)). Thus, the probability of $\boldsymbol{x}_{\ell(h)}$ being strictly preferred to $\boldsymbol{x}_{\kappa(h)}$ is:

$$\begin{aligned} p\left(\boldsymbol{x}_{\ell(h)} \succ \boldsymbol{x}_{\kappa(h)} \,\middle|\, f_{\ell(h)}, f_{\kappa(h)}, \boldsymbol{x}_{\ell(h)}, \boldsymbol{x}_{\kappa(h)}\right) &= p\left(y_{\ell(h)} < y_{\kappa(h)} \,\middle|\, f_{\ell(h)}, f_{\kappa(h)}, \boldsymbol{x}_{\ell(h)}, \boldsymbol{x}_{\kappa(h)}\right) \\ &= p\left(\eta_{f_{\ell(h)}} - \eta_{f_{\kappa(h)}} < f_{\kappa(h)} - f_{\ell(h)} \,\middle|\, f_{\ell(h)}, f_{\kappa(h)}, \boldsymbol{x}_{\ell(h)}, \boldsymbol{x}_{\kappa(h)}\right). \end{aligned}$$

Notice that $\eta_{f_{\ell(h)}} - \eta_{f_{\kappa(h)}}$ is a zero-mean normally distributed random variable with variance $2 \cdot \sigma_f^2$. Therefore, by standardizing $\eta_{f_{\ell(h)}} - \eta_{f_{\kappa(h)}}$, we obtain:

$$\begin{aligned} p\left(\boldsymbol{x}_{\ell(h)} \succ \boldsymbol{x}_{\kappa(h)} \,\middle|\, f_{\ell(h)}, f_{\kappa(h)}, \boldsymbol{x}_{\ell(h)}, \boldsymbol{x}_{\kappa(h)}\right) &= p\left(\frac{\eta_{f_{\ell(h)}} - \eta_{f_{\kappa(h)}}}{\sqrt{2} \cdot \sigma_f} < \frac{f_{\kappa(h)} - f_{\ell(h)}}{\sqrt{2} \cdot \sigma_f} \,\middle|\, f_{\ell(h)}, f_{\kappa(h)}, \boldsymbol{x}_{\ell(h)}, \boldsymbol{x}_{\kappa(h)}\right) \\ &= \Phi_{\mathcal{N}}\left(\frac{f_{\kappa(h)} - f_{\ell(h)}}{\sqrt{2} \cdot \sigma_f}\right). \end{aligned}$$

Thus, the preference information brought by the set $\mathcal{S}$ in (3.10) is encapsulated in the following *likelihood*:

$$\begin{aligned} p\left(\mathcal{S} \,\middle|\, \boldsymbol{f}, \mathcal{X}\right) &= \prod_{h=1}^{M} p\left(\boldsymbol{x}_{\ell(h)} \succ \boldsymbol{x}_{\kappa(h)} \,\middle|\, f_{\ell(h)}, f_{\kappa(h)}, \boldsymbol{x}_{\ell(h)}, \boldsymbol{x}_{\kappa(h)}\right) \\ &= \prod_{h=1}^{M} \Phi_{\mathcal{N}}\left(\frac{f_{\kappa(h)} - f_{\ell(h)}}{\sqrt{2} \cdot \sigma_f}\right). \end{aligned} \tag{3.14}$$

Next, we need to compute the *posterior distribution* $p\left(\boldsymbol{f} \,\middle|\, \mathcal{S}, \mathcal{X}; \boldsymbol{\theta}_f\right)$. However, this time, the posterior distribution is not Gaussian since the likelihood in (3.14) is not Gaussian. In [25], the authors find the *Laplace approximation* [15] of $p\left(\boldsymbol{f} \,\middle|\, \mathcal{S}, \mathcal{X}; \boldsymbol{\theta}_f\right)$ as follows. From Bayes' theorem, we have:

$$\begin{aligned} p\left(\boldsymbol{f} \,\middle|\, \mathcal{S}, \mathcal{X}; \boldsymbol{\theta}_f\right) &= \frac{p\left(\mathcal{S} \,\middle|\, \boldsymbol{f}, \mathcal{X}\right) \cdot p\left(\boldsymbol{f}; \boldsymbol{\theta}_f\right)}{p\left(\mathcal{S} \,\middle|\, \mathcal{X}; \boldsymbol{\theta}_f\right)} \\ &\propto p\left(\mathcal{S} \,\middle|\, \boldsymbol{f}, \mathcal{X}\right) \cdot p\left(\boldsymbol{f}; \boldsymbol{\theta}_f\right). \end{aligned}$$

Define $\tilde{s}\left(\boldsymbol{f}; \boldsymbol{\theta}_f\right) = p\left(\mathcal{S} \,\middle|\, \boldsymbol{f}, \mathcal{X}\right) \cdot p\left(\boldsymbol{f}; \boldsymbol{\theta}_f\right)$ and take its natural logarithm:

$$\begin{aligned} \ln \tilde{s}\left(\boldsymbol{f}; \boldsymbol{\theta}_f\right) = &\sum_{h=1}^{M} \ln\left[\Phi_{\mathcal{N}}\left(\frac{f_{\kappa(h)} - f_{\ell(h)}}{\sqrt{2} \cdot \sigma_f}\right)\right] + \ln\left[\frac{1}{\sqrt{(2 \cdot \pi)^N \cdot \det K_f\left(\boldsymbol{\theta}_f\right)}}\right] + \\ &- \frac{1}{2} \cdot \boldsymbol{f}^\top \cdot K_f\left(\boldsymbol{\theta}_f\right)^{-1} \cdot \boldsymbol{f}. \end{aligned}$$

Now, let $s\left(\boldsymbol{f}; \boldsymbol{\theta}_f\right) = -\ln \tilde{s}\left(\boldsymbol{f}; \boldsymbol{\theta}_f\right)$, neglecting the term that does not depend on $\boldsymbol{f}$, i.e.:

$$s\left(\boldsymbol{f}; \boldsymbol{\theta}_f\right) = \frac{1}{2} \cdot \boldsymbol{f}^\top \cdot K_f\left(\boldsymbol{\theta}_f\right)^{-1} \cdot \boldsymbol{f} - \sum_{h=1}^{M} \ln\left[\Phi_{\mathcal{N}}\left(\frac{f_{\kappa(h)} - f_{\ell(h)}}{\sqrt{2} \cdot \sigma_f}\right)\right].$$

We compute the *Maximum A Posteriori (MAP)* estimate of $\boldsymbol{f}$ in (2.36) by solving the following optimization problem:

$$\begin{aligned} \boldsymbol{f}_{MAP} &= \arg \max_{\boldsymbol{f} \in \mathbb{R}^N} p\left(\boldsymbol{f} \,\middle|\, \mathcal{S}, \mathcal{X}; \boldsymbol{\theta}_f\right) \\ &= \arg \max_{\boldsymbol{f} \in \mathbb{R}^N} \ln \tilde{s}\left(\boldsymbol{f}; \boldsymbol{\theta}_f\right) \\ &= \arg \min_{\boldsymbol{f} \in \mathbb{R}^N} s\left(\boldsymbol{f}; \boldsymbol{\theta}_f\right). \end{aligned} \tag{3.15}$$

Problem (3.15) is convex [25]; furthermore, the Hessian of $s\left(\boldsymbol{f}; \boldsymbol{\theta}_f\right)$ is equal to:

$$\nabla^2_{\boldsymbol{f}\boldsymbol{f}} s\left(\boldsymbol{f}; \boldsymbol{\theta}_f\right) = K_f\left(\boldsymbol{\theta}_f\right)^{-1} + \Lambda\left(\boldsymbol{f}\right),$$

where $\Lambda\left(\boldsymbol{f}\right) = -\sum_{h=1}^{M} \nabla^2_{\boldsymbol{f}\boldsymbol{f}} \ln \Phi_{\mathcal{N}}\left(\frac{f_{\kappa(h)} - f_{\ell(h)}}{\sqrt{2} \cdot \sigma_f}\right)$. Finally, we approximate the posterior distribution as:

$$p\left(\boldsymbol{f} \,\middle|\, \mathcal{S}, \mathcal{X}; \boldsymbol{\theta}_f\right) \approx \mathcal{N}\left(\boldsymbol{\mu}_{\boldsymbol{f} | \mathcal{S}, \mathcal{X}}, \Sigma_{\boldsymbol{f} | \mathcal{S}, \mathcal{X}}\right), \tag{3.16a}$$

$$\boldsymbol{\mu}_{\boldsymbol{f} | \mathcal{S}, \mathcal{X}} = \boldsymbol{f}_{MAP}, \tag{3.16b}$$

$$\Sigma_{\boldsymbol{f} | \mathcal{S}, \mathcal{X}} = \left[K_f\left(\boldsymbol{\theta}_f\right)^{-1} + \Lambda\left(\boldsymbol{f}_{MAP}\right)\right]^{-1}, \tag{3.16c}$$

and the *marginal likelihood* as:

$$p\left(\mathcal{S} \,\middle|\, \boldsymbol{\theta}_f\right) \approx \exp\left\{-s\left(\boldsymbol{f}_{MAP}\right)\right\} \cdot \frac{1}{\sqrt{\det\left[I_N + K_f\left(\boldsymbol{\theta}_f\right) \cdot \Lambda\left(\boldsymbol{f}_{MAP}\right)\right]}}. \tag{3.17}$$

The latter can be used to estimate the hyper-parameters $\boldsymbol{\theta}_f$ of the kernel through maximum likelihood estimation, i.e. by solving Problem (2.41).

Similarly to Gaussian process regression (in Section 2.5.2), we are interested in predicting the value of the scoring function $f(\boldsymbol{x})$ of the GOP (2.1) at a point that has not been previously evaluated, i.e. we want to estimate $\tilde{f} = f(\tilde{\boldsymbol{x}})$ for some $\tilde{\boldsymbol{x}} \notin \mathcal{X}$. To do so, we define the prior on the random vector $\begin{bmatrix}\boldsymbol{f}^\top & \tilde{f}\end{bmatrix}^\top$ as in (3.13). Then, we find the joint posterior distribution $p\left(\boldsymbol{f}, \tilde{f} \,\middle|\, \mathcal{S}, \mathcal{X}, \tilde{\boldsymbol{x}}; \boldsymbol{\theta}_f\right)$ by considering the Laplace approximation of $p\left(\boldsymbol{f} \,\middle|\, \mathcal{S}, \mathcal{X}; \boldsymbol{\theta}_f\right)$ in (3.16), which is Gaussian. Lastly, we marginalize with respect to $\tilde{f}$, obtaining the *predictive distribution*:

$$p\left(\tilde{f} \,\middle|\, \boldsymbol{f}, \mathcal{S}, \mathcal{X}, \tilde{\boldsymbol{x}}; \boldsymbol{\theta}_f\right) \approx \mathcal{N}\left(\mu_{\tilde{f}|\boldsymbol{f},\mathcal{S},\mathcal{X},\tilde{\boldsymbol{x}}}, \Sigma_{\tilde{f}|\boldsymbol{f},\mathcal{S},\mathcal{X},\tilde{\boldsymbol{x}}}\right), \tag{3.18a}$$

$$\mu_{\tilde{f}|\boldsymbol{f},\mathcal{S},\mathcal{X},\tilde{\boldsymbol{x}}} = \boldsymbol{k}_f\left(\tilde{\boldsymbol{x}}; \boldsymbol{\theta}_f\right)^\top \cdot K_f\left(\boldsymbol{\theta}_f\right)^{-1} \cdot \boldsymbol{f}_{MAP}, \tag{3.18b}$$

$$\Sigma_{\tilde{f}|\boldsymbol{f},\mathcal{S},\mathcal{X},\tilde{\boldsymbol{x}}} = k_f\left(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{x}}; \boldsymbol{\theta}_f\right) - \boldsymbol{k}_f\left(\tilde{\boldsymbol{x}}; \boldsymbol{\theta}_f\right)^\top \cdot \left[K_f\left(\boldsymbol{\theta}_f\right) + \Lambda\left(\boldsymbol{f}_{MAP}\right)^{-1}\right]^{-1} \cdot \boldsymbol{k}_f\left(\tilde{\boldsymbol{x}}; \boldsymbol{\theta}_f\right), \tag{3.18c}$$

where $\boldsymbol{k}_f\left(\tilde{\boldsymbol{x}}; \boldsymbol{\theta}_f\right)$ is defined as in (2.45). Finally, the surrogate model of the scoring function $f(\boldsymbol{x})$ of the GOP (2.1) is the mean of the predictive distribution:

$$\hat{f}_N\left(\boldsymbol{x}; \boldsymbol{\beta}_f, \boldsymbol{\theta}_f\right) = \boldsymbol{k}_f\left(\boldsymbol{x}; \boldsymbol{\theta}_f\right)^\top \cdot \boldsymbol{\beta}_f, \tag{3.19}$$

where $\boldsymbol{\beta}_f = K_f\left(\boldsymbol{\theta}_f\right)^{-1} \cdot \boldsymbol{f}_{MAP}$.

As an alternative to the surrogate scoring function proposed in [25], recently the authors of [12] have derived an approximation of $f(\boldsymbol{x})$ based on Skew Gaussian processes. One of the advantages of the method in [12] is that the posterior distribution can be computed in closed form, without the need for the Laplace approximation.

3.4.3 Surrogate preference function based on $\mathcal{GP}$s

In a recent article on preference-based optimization [52], the authors propose three acquisition functions that are not based on a surrogate of the scoring function but rather on a probabilistic model of the preference relation $\gtrsim$ on Ω. In particular, instead of considering the deterministic preference function $\pi_{\gtrsim}\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right)$ in (3.8), they model the probability of a sample $\boldsymbol{x}_i \in \Omega$ being preferred to another $\boldsymbol{x}_j \in \Omega$ using a Bernoulli distribution:

$$p\left(\boldsymbol{x}_i \succ \boldsymbol{x}_j \,\middle|\, f\left(\boldsymbol{x}_i\right), f\left(\boldsymbol{x}_j\right), \boldsymbol{x}_i, \boldsymbol{x}_j\right) = \tilde{\pi}_{\gtrsim}\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right), \tag{3.20a}$$

$$p\left(\boldsymbol{x}_j \gtrsim \boldsymbol{x}_i \,\middle|\, f\left(\boldsymbol{x}_i\right), f\left(\boldsymbol{x}_j\right), \boldsymbol{x}_i, \boldsymbol{x}_j\right) = 1 - \tilde{\pi}_{\gtrsim}\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right). \tag{3.20b}$$

In (3.20), $f(\cdot)$ is the scoring function and $\tilde{\pi}_{\gtrsim} : \mathbb{R}^n \times \mathbb{R}^n \to [0, 1]$ is an *inverse link function*. A typical choice is the *sigmoid* function:

$$\tilde{\pi}_{\gtrsim}\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right) = \frac{1}{1 + \exp\left\{f_{\gtrsim}\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right)\right\}}, \tag{3.21}$$

where $f_{\gtrsim} : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ is simply the difference between the two scores, namely $f_{\gtrsim}\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right) = f\left(\boldsymbol{x}_i\right) - f\left(\boldsymbol{x}_j\right)$. Intuitively, if $\boldsymbol{x}_i \succ \boldsymbol{x}_j$, then $f\left(\boldsymbol{x}_i\right) < f\left(\boldsymbol{x}_j\right)$ and $f_{\gtrsim}\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right) < 0$, making $\tilde{\pi}_{\gtrsim}\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right) > \frac{1}{2}$. The relationship between the preference function $\pi_{\gtrsim}\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right)$ in (3.8) and $\tilde{\pi}_{\gtrsim}\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right)$ in (3.21) is the following:

$$\pi_{\gtrsim}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \begin{cases} -1 & \text{if } \tilde{\pi}_{\gtrsim}\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right) > \frac{1}{2} \\ 0 & \text{if } \tilde{\pi}_{\gtrsim}\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right) = \frac{1}{2} \\ 1 & \text{if } \tilde{\pi}_{\gtrsim}\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right) < \frac{1}{2} \end{cases} .$$

A surrogate model for the probabilistic preference function in (3.21) can be obtained by means of Gaussian process classification. Here, we forego an in-depth dissertation, the interested reader is referred [52, 60, 153]. Intuitively, given two samples $\boldsymbol{x}_i, \boldsymbol{x}_j \in \Omega$, we can define two classes as in (3.20), which we call the "is preferred to" class ($\boldsymbol{x}_i \succ \boldsymbol{x}_j$, $f_{\gtrsim}\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right) < 0$) and the "is not preferred to" class ($\boldsymbol{x}_j \succ \boldsymbol{x}_i$, $f_{\gtrsim}\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right) \geq 0$). The basic idea behind Gaussian process classification is to place a $\mathcal{GP}$ prior over the latent $f_{\gtrsim}(\cdot, \cdot)$, which is considered as a generic function in $2 \cdot n$ variables that captures the membership of the data to the two classes. Then, similarly to Section 3.4.2, we obtain the predictive distribution of $f_{\gtrsim}(\cdot, \cdot)$, namely

$$p\left(\tilde{f}_{\gtrsim} \,\middle|\, \boldsymbol{f}_{\gtrsim}, \mathcal{S}, \mathcal{X}, \tilde{\boldsymbol{x}}_1, \tilde{\boldsymbol{x}}_2; \boldsymbol{\theta}_{f_{\gtrsim}}\right),$$

where $\tilde{f}_{\gtrsim} = f_{\gtrsim}(\tilde{\boldsymbol{x}}_1, \tilde{\boldsymbol{x}}_2)$, $\tilde{\boldsymbol{x}}_1, \tilde{\boldsymbol{x}}_2 \notin \mathcal{X}$, $\boldsymbol{f}_{\gtrsim}$ is the vector of function evaluations of $f_{\gtrsim}(\cdot, \cdot)$ (analogously to (2.36)) and $\boldsymbol{\theta}_{f_{\gtrsim}}$ is a vector of hyper-parameters for the chosen kernel. Lastly, the surrogate model for the probabilistic preference function in (3.21), namely $\hat{\tilde{\pi}}_{\gtrsim_N} : \mathbb{R}^n \times \mathbb{R}^n \to [0, 1]$, is obtained by "squashing" the $\mathcal{GP}$ predictions of $f_{\gtrsim}(\cdot, \cdot)$ through the sigmoid function in (3.21). More formally:

$$\begin{aligned} \hat{\tilde{\pi}}_{\gtrsim_N}\left(\tilde{\boldsymbol{x}}_1 \succ \tilde{\boldsymbol{x}}_2; \boldsymbol{\theta}_{f_{\gtrsim}}\right) &= p\left(\tilde{\boldsymbol{x}}_1 \succ \tilde{\boldsymbol{x}}_2 \,\middle|\, \boldsymbol{f}_{\gtrsim}, \mathcal{S}, \mathcal{X}, \tilde{\boldsymbol{x}}_1, \tilde{\boldsymbol{x}}_2; \boldsymbol{\theta}_{f_{\gtrsim}}\right) \\ &= \int \frac{1}{1 + \exp\left\{\tilde{f}_{\gtrsim}\right\}} \cdot p\left(\tilde{f}_{\gtrsim} \,\middle|\, \boldsymbol{f}_{\gtrsim}, \mathcal{S}, \mathcal{X}, \tilde{\boldsymbol{x}}_1, \tilde{\boldsymbol{x}}_2; \boldsymbol{\theta}_{f_{\gtrsim}}\right) d\tilde{f}_{\gtrsim}. \end{aligned} \tag{3.22}$$

In general, the computation of the posterior $p\left(\tilde{f}_{\gtrsim} \,\middle|\, \boldsymbol{f}_{\gtrsim}, \mathcal{S}, \mathcal{X}, \tilde{\boldsymbol{x}}_1, \tilde{\boldsymbol{x}}_2; \boldsymbol{\theta}_{f_{\gtrsim}}\right)$ and of the integral in (3.22) might be analytically intractable, hence suitable approximations are needed [153].

3.5 Infill sampling criteria

Similarly to black-box optimization, surrogate-based methods for preference-based optimization look for new candidate samples using proper infill sampling criteria, following the rationale described in Section 2.6. The new point $\boldsymbol{x}_{N+1} \in \Omega$ is found by solving an additional global optimization problem, such as Problem (2.48a) or Problem (2.48b). In the preference-based framework, the acquisition functions rely on a surrogate model of the latent scoring function instead of an approximation of the black-box cost function of the GOP (2.1). Once $\boldsymbol{x}_{N+1}$ has been found, we ask the decision-maker to compare it to the current best candidate, $\boldsymbol{x_{best}}(N)$, obtaining the preference:

$$b_{M+1} = \pi_{\gtrsim}\left(\boldsymbol{x}_{N+1}, \boldsymbol{x_{best}}(N)\right).$$

The black-box constraints at $\boldsymbol{x}_{N+1}$ are also assessed (if any are present). Alternatively, there exist PBO procedures whose infill sampling criteria actually return more than one candidate sample for evaluation. For example, in [52] the authors propose three different strategies that return two points, $\boldsymbol{x}_{N+1}, \boldsymbol{x}_{N+2} \in \Omega$, among which the DM is asked to express a preference.

As of now, in the literature there exist very few articles that derive surrogate-based PBO procedures, when compared to their black-box counterpart. Most of them only tackle the unconstrained preference-based optimization case, although it is straightforward to extend the methods to the constrained PBO framework since the black-box constraints can be handled in the same way as in constrained BBO (see Section 2.6). On a side note, *in preference-based optimization, the black-box constraints might be handled implicitly by the decision-maker*. For example, if Ξ in (2.2) includes only DM-based constraints (Definition 3.6) and the decision-maker who assess the Ξ-feasibility is also the one expressing the preferences, then surely a Ξ-feasible sample is preferred to any Ξ-infeasible one.

The most popular surrogate-based PBO algorithms follow the Bayesian Optimization rationale, adapted to the case where only preference information is available. In what follows, we refer to the latter as Preferential Bayesian Optimization (PrefBayesOpt) [52] methods.

The goal of finding the most preferred choice (of a decision-maker) among a set of alternatives has also been tackled in the multi-armed bandit literature. In particular, the survey in [13] presents an exhaustive review of different active learning algorithms for multi-armed dueling bandit problems. Typically, these methods assume that the DM can only express preferences among a finite number of options. Furthermore, the black-box constraints are not considered. In this book, we only focus on surrogate-based methods and now review the most popular infill sampling criteria in the preference-based setting.

In what follows, we assume to have performed $N \in \mathbb{N}$ sample evaluations, resulting in $1 \leq M \leq \begin{pmatrix} N \\ 2 \end{pmatrix}$ preferences and N black-box constraints assessments, as described by the sets $\mathcal{X}$ in (2.9), $\mathcal{B}$ in (3.9), $\mathcal{S}$ in (3.10), $\mathcal{U}_{\Xi}$ in (2.11) and (possibly but not necessarily) $\mathcal{C}_{\Xi}$ in (2.12). Similarly to Section 2.6, we use a more concise notation for the surrogates, refraining from indicating their hyper-parameters.

3.5.1 Infill sampling criteria for methods based on deterministic surrogates

`GLISp` [11]. `GLISp` [11] is an extension of the black-box optimization algorithm `GLIS` [10] (reviewed in Section 2.6.1) to the unconstrained preference-based framework. The method uses the surrogate scoring function $\hat{f}_N(\boldsymbol{x})$ in (2.20), which is based on radial basis functions, with the vector of weights $\boldsymbol{\beta}_f$ computed as in Section 3.4.1. The acquisition function $a_N(\boldsymbol{x})$ of `GLISp` [11] is quite similar to that of `GLIS` [10], in (2.58), but the IDW variance function, $s_N(\boldsymbol{x})$ in (2.56b), is omitted since the measures of $f(\boldsymbol{x})$ are not available in the preference-based setting. Hence, $a_N(\boldsymbol{x})$ of `GLISp` [11] amounts to:

$$a_N(\boldsymbol{x}) = \frac{\hat{f}_N(\boldsymbol{x})}{\Delta \hat{F}} + \delta \cdot z_N(\boldsymbol{x}), \tag{3.23}$$

where $\delta \in \mathbb{R}_{\geq 0}$ is a (fixed) exploration-exploitation trade-off weight that is selected by the user, $z_N(\boldsymbol{x})$ is the IDW distance function in (2.56a) and $\Delta \hat{F} = \max_{\boldsymbol{x}_i \in \mathcal{X}} \hat{f}_N(\boldsymbol{x}_i) - \min_{\boldsymbol{x}_i \in \mathcal{X}} \hat{f}_N(\boldsymbol{x}_i)$. The new candidate sample $\boldsymbol{x}_{N+1}$ is found by minimizing $a_N(\boldsymbol{x})$ in (3.23), i.e. by solving Problem (2.48a). `GLISp` [11] has been extended to the constrained preference-based optimization framework, giving rise to `C-GLISp` [156], which we now cover.

`C-GLISp` [156]. In [156], the authors consider two separate decision-maker-based constraints: acceptability (which they refer to as "feasibility") and satisfaction. In practice, this is the same as considering two separate sets, Ξ_A and Ξ_S for the two criteria respectively, and defining the black-box constraint set as $\Xi = \Xi_A \cap \Xi_S$. Here, we stay consistent to our definition of the GOP (2.1) and simply consider a sample to be Ξ-feasible whenever it is both acceptable and satisfactory. `C-GLISp` [156] follows a rationale that is similar to that of Bayesian optimization algorithms in the constrained BBO framework: at each iteration, an approximation of the probability of Ξ-feasibility, $p_N(\boldsymbol{x} \in \Xi)$, is derived from the data at hand; the latter is then used to penalize some of the regions of Ω when looking for the next candidate sample $\boldsymbol{x}_{N+1}$. $p_N(\boldsymbol{x} \in \Xi)$ is approximated using an interpolation method, called inverse distance weighting interpolation, obtaining:

$$\begin{aligned} p_N (\boldsymbol{x} \in \Xi) &= p\left(\boldsymbol{x} \in \Xi \,\middle|\, \mathcal{U}_\Xi, \mathcal{X}, \boldsymbol{x}\right) \\ &= \sum_{i=1}^{N} v_i (\boldsymbol{x}) \cdot u_i. \end{aligned} \tag{3.24}$$

In (3.24), $u_i \in \{0, 1\}$ is the feasibility information contained inside the set $\mathcal{U}_\Xi$ in (2.11), while $v : \mathbb{R}^n \to \mathbb{R}$ is defined as in (2.57). The acquisition function for C-GLISp [156] is[2]:

$$a_N (\boldsymbol{x}) = \frac{\hat{f}_N (\boldsymbol{x})}{\Delta \hat{F}} + \delta \cdot z_N (\boldsymbol{x}) + \delta_\Xi \cdot [1 - p_N (\boldsymbol{x} \in \Xi)] \,. \tag{3.25}$$

In (3.25), $\delta, \delta_\Xi \in \mathbb{R}_{\geq 0}$ are the weights for the exploration and penalization terms respectively. Moreover, $z_N (\boldsymbol{x})$ in (3.25) is a revisitation of the IDW exploration function in (2.56a) that is better suited for escaping local minima of the GOP (2.1).

We point out that, *as of now, to the best of our knowledge, C-GLISp [156] is the only preference-based optimization algorithm in the literature which also handles black-box constraints (that are assumed to be DM-based constraints).* The same way of penalizing the black-box constraints violations has also been implemented in C-GLIS, which is used for constrained black-box optimization. C-GLIS is an extension of GLIS [10] that relies on an acquisition function defined as the sum of $a_N (\boldsymbol{x})$ in (2.58) (unconstrained BBO) and the penalty term $\delta_\Xi \cdot [1 - p_N (\boldsymbol{x} \in \Xi)]$ in (3.25). Formally, there are no articles that describe the latter method, although it is supplied in the same software package of C-GLISp [156].

GLIS [10], GLISp [11], C-GLIS and C-GLISp [156] will be covered in greater detail in Chapter 4. Furthermore, in Chapter 5 and Chapter 6, we will propose an extension for each method.

3.5.2 Infill sampling criteria for preferential Bayesian optimization methods

PrefBayesOpt with scoring function surrogates. The definition of acquisition functions for preferential Bayesian optimization procedures based on the surrogate scoring function described in Section 3.4.2 is straightforward. In practice, all the criteria that we have seen in Section 2.6.2 for BayesOpt, namely the probability of improvement in (2.64), the expected improvement in (2.67) and the lower confidence bound in (2.68), are still valid for PBO, provided that we use the mean and the variance of the predictive distribution in (3.18) instead of those in (2.44). Moreover, the measure of the black-box cost function at $\boldsymbol{x}_{\boldsymbol{best}} (N)$, i.e. $y_{best} (N)$, is not available in the preference-based setting and it is

[2] In [156], the authors approximate both the probability of acceptability and the probability of satisfiability, including two penalization terms inside $a_N (\boldsymbol{x})$ instead of only $p_N (\boldsymbol{x} \in \Xi)$.

replaced by:

$$y_{best}(N) = \arg \min_{\boldsymbol{x}_i \in \mathcal{X}} \hat{f}_N(\boldsymbol{x}_i)$$

in the aforementioned acquisition functions. One of the first surrogate-based PrefBayesOpt algorithms is presented in [21]. The latter method uses $a_N(\boldsymbol{x})$ in (2.67) to drive the search towards those samples that are the most preferred by the human decision-maker. Notably, in [12], the authors define the "dueling" versions of some popular infill sampling criteria for BayesOpt, which are acquisition functions based on the difference $f(\boldsymbol{x}) - f(\boldsymbol{x_{best}}(N))$ instead of only $f(\boldsymbol{x})$.

PrefBayesOpt with preference function surrogates. Next, we consider the preference-based optimization algorithm proposed in [52], which relies on the probabilistic interpretation of the preference function $\tilde{\pi}_{\gtrsim}(\cdot,\cdot)$ in (3.20) (Section 3.4.3). In this setting, the global minimizers of the GOP (2.1) can be sought by maximizing the *soft-Copeland score*:

$$c_{\gtrsim}(\boldsymbol{x}) = \mathrm{Vol}(\Omega)^{-1} \cdot \int_{\Omega} \tilde{\pi}_{\gtrsim}(\boldsymbol{x}, \tilde{\boldsymbol{x}})\, d\tilde{\boldsymbol{x}}, \tag{3.26}$$

where $\mathrm{Vol}(\Omega)^{-1} = \int_{\Omega} d\tilde{\boldsymbol{x}}$ is a normalization constant. $c_{\gtrsim}(\boldsymbol{x})$ in (3.26) is the "averaged" probability of $\boldsymbol{x}$ being preferred over all the other samples in Ω. We say that a sample $\boldsymbol{x} \in \Omega$ is a *Condorcet winner* if its corresponding soft-Copeland score is maximal. In practice, any Condorcet winner is a solution of the GOP (2.1).

The authors of [52] propose three different infill sampling criteria, all based on the surrogate preference function $\hat{\tilde{\pi}}_{\gtrsim_N}(\cdot,\cdot)$ in (3.22). Here, we only review the best performing one, called Dueling-Thompson sampling, which works as follows. At each iteration, the algorithm produces two candidate samples (instead of just one), $\boldsymbol{x}_{N+1}, \boldsymbol{x}_{N+2} \in \Omega$, among which the decision-maker is asked to state his/her preference:

1. The first point is selected by maximizing the soft-Copeland score in (3.26), using the surrogate $\hat{\tilde{\pi}}_{\gtrsim_N}(\cdot,\cdot)$ in (3.22) instead of $\tilde{\pi}_{\gtrsim}(\cdot,\cdot)$ in (3.21), i.e.:

$$\begin{aligned} \boldsymbol{x}_{N+1} = \arg \max_{\boldsymbol{x}} & \int_{\Omega} \hat{\tilde{\pi}}_{\gtrsim_N}(\boldsymbol{x}, \tilde{\boldsymbol{x}})\, d\tilde{\boldsymbol{x}} \\ \text{s.t.} \quad & \boldsymbol{x} \in \Omega. \end{aligned} \tag{3.27}$$

   In particular, $\boldsymbol{x}_{N+1}$ in (3.27) is selected through Thompson sampling [122].

2. The second sample is selected as the one for which the output of the comparison between $\boldsymbol{x}_{N+1}$ and $\boldsymbol{x}_{N+2}$ is the most uncertain. To do so, we maximize the variance of the sigmoid function

$s\left(\tilde{f}_{\gtrsim}\right) = \frac{1}{1+\exp\{\tilde{f}_{\gtrsim}\}}$ in (3.22), keeping the first sample fixed. Formally, the resulting acquisition function is:

$$a_{N+1}(\boldsymbol{x}) = \int \left\{s\left(\tilde{f}_{\gtrsim}\right) - \mathbb{E}\left[s\left(\tilde{f}_{\gtrsim}\right)\right]\right\}^2 \cdot p\left(\tilde{f}_{\gtrsim} \,\middle|\, \boldsymbol{f}_{\gtrsim}, \mathcal{S}, \mathcal{X}, \boldsymbol{x}_{N+1}, \boldsymbol{x}; \boldsymbol{\theta}_{f_{\gtrsim}}\right) d\tilde{f}_{\gtrsim}.$$

In practice, the latter integral is approximated by Monte Carlo integration.

Overall, the Dueling-Thompson sampling criterion selects a point $\boldsymbol{x}_{N+1}$ that is likely to be a Condorcet winner (exploitation) and a sample $\boldsymbol{x}_{N+2}$ for which the result of the comparison with the former is the most uncertain (exploration).

3.6 Summary of global, black-box and preference-based optimization

This last Section is devoted to summarizing what we have seen so far on global (Chapter 1), black-box (Chapter 2) and preference-based (Chapter 3) optimization. Regardless of the framework, it all boils down to solving the following global optimization problem:

$$\begin{aligned} \mathcal{X}^* &= \arg\min_{\boldsymbol{x}} f(\boldsymbol{x}) \\ \text{s.t.} &\quad \boldsymbol{x} \in \Omega \cap \Xi, \end{aligned}$$

where Ω is a set of completely known constraints (most often, bounds on the decision variables). *Global, black-box and preference-based optimization differ on the information available on the cost function $f(\boldsymbol{x})$ and the constraints functions in Ξ, as well as on how the samples are evaluated* (see Definition 1.1). In global optimization, we assume that the analytical expressions of $f(\boldsymbol{x})$ and of all the constraints functions are available ($\Xi = \mathbb{R}^n$). A sample evaluation in GO simply amounts to computing the latter functions at a given point. Instead, in black-box and preference-based optimization, we assume that information on $f(\boldsymbol{x})$ and (possibly but not necessarily) on the constraints in Ξ can only be acquired by running time-consuming computer simulations or performing expensive real-world experiments. Even though we do not know a-priori the mathematical relationship between the decision vector $\boldsymbol{x}$ and the cost function $f(\boldsymbol{x})$, in BBO we assume to be able to measure the latter at any given point $\boldsymbol{x}_i \in \Omega$, obtaining $y_i = f(\boldsymbol{x}_i)$. The measures could also be affected by noise although, in this book, we restrict ourselves to the noiseless case (see Assumption 2.4). Instead, in preference-based optimization, $f(\boldsymbol{x})$ cannot be measured in any way: the optimization must be carried out by relying only on the preferences expressed by a human decision-maker. From an utility theory standpoint, we have seen how finding the calibration that is most preferred by a DM is equivalent to minimizing a latent scoring function, $f(\boldsymbol{x})$, that ranks the samples in Ω as the decision-maker would (see Section 3.1 and Section 3.2). Lastly, whenever black-box constraints are present ($\Xi \subset \mathbb{R}^n$), both in BBO and in PBO we assume that, given a sample $\boldsymbol{x}_i \in \Omega$, we can either measure the constraints functions $\boldsymbol{g}_\Xi(\cdot)$ in (2.2) at $\boldsymbol{x}_i$ or, at least, assess if the point is Ξ-feasible or not through the Ξ-feasibility function $u_\Xi(\cdot)$ in (2.4). In preference-based optimization, we have also highlighted a particular type of black-box constraints, i.e. decision-maker-based constraints (see Definition 3.6), which can be seen as asking the DM a "yes/no question" (such as "is $\boldsymbol{x}_i$ acceptable?").

In summary, *we can view black-box optimization as a specific instance of global optimization* where the cost function and (possibly but not necessarily) some of the constraints functions are unknown (black-boxes) but can be measured in some way. Similarly, *preference-based optimization is a particular case of BBO* wherein $f(\boldsymbol{x})$ is a black-box whose values are inaccessible. Instead, information on the scoring function comes only in the form of comparisons between samples made by the human decision-maker.

The *exploration-exploitation dilemma* is a recurrent topic in global, black-box and preference-based optimization. Exploiting means to take advantage of the information acquired on the global optimization problem at hand to drive the search towards its global solution(s). Instead, exploring simply means to probe the feasible region $\Omega \cap \Xi$ in the hope of finding more promising zones that have yet to be examined. In global optimization, we have seen that there are several ways of tackling the exploration-exploitation dilemma (cf. the taxonomy in Section 1.2). In particular, for what concerns black-box and preference-based optimization, the de facto standard procedures are *surrogate-based methods* (or response surface techniques, see Section 2.2), whose main objective is to minimize the number of sample evaluations required to find a "sufficiently optimal" solution (at the cost of higher computational times). In surrogate-based methods, the exploration-exploitation dilemma is addressed by *infill sampling criteria*, of which there are plenty (see Section 2.6 and Section 3.5).

Finally, we conclude this Chapter with Figure 9, which sums up the three optimization frameworks and reports all the algorithms that we have reviewed so far in this book. Moreover, we also include the procedures that we will propose in Chapter 5 and Chapter 6, highlighting to which framework they belong.

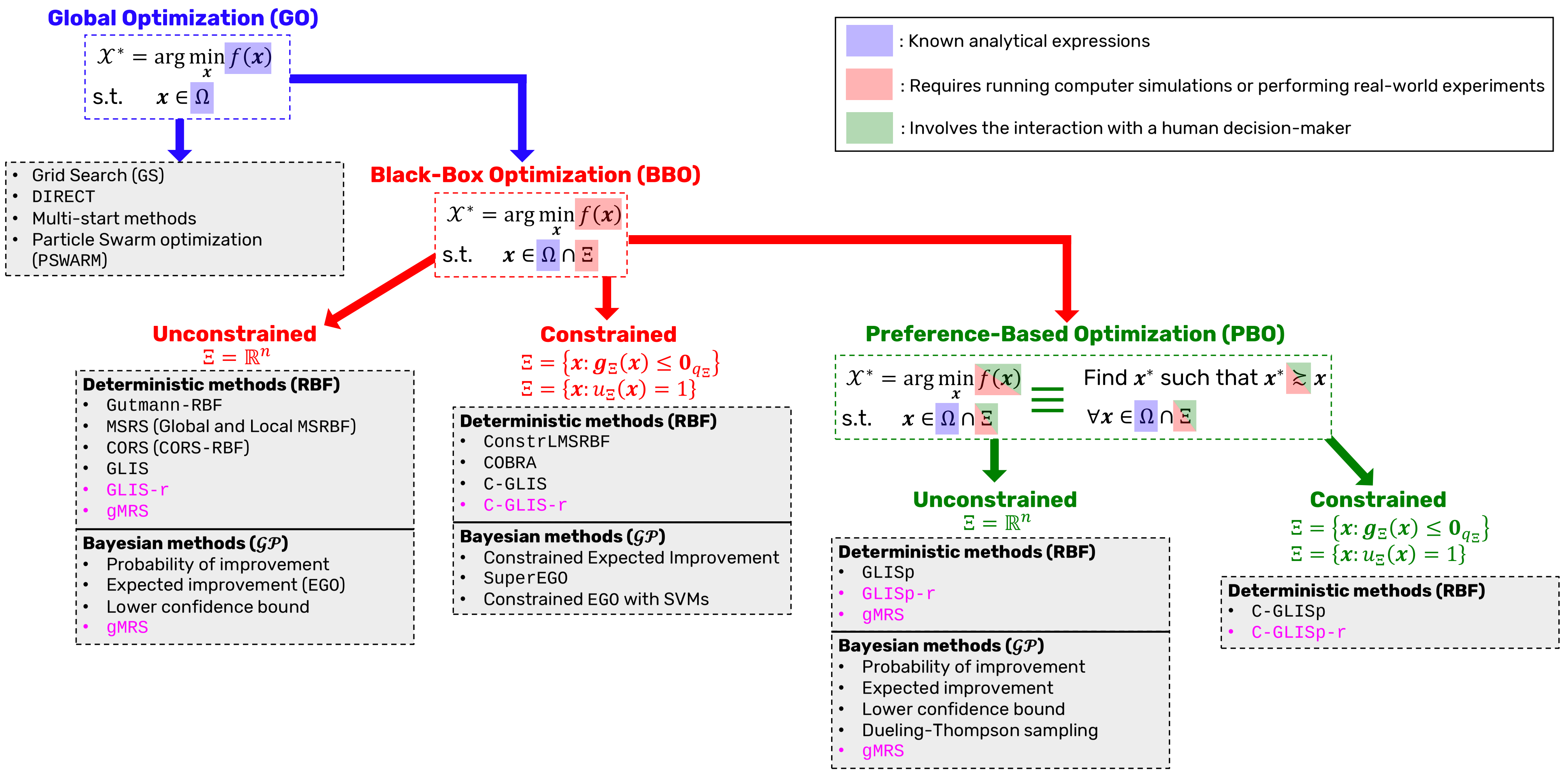

Global Optimization (GO)
$\mathcal{X}^* = \arg\min_{x} f(x)$
s.t. $x \in \Omega$
Grid Search (GS)
DIRECT
Multi-start methods
Particle Swarm optimization (PSWARM)
Black-Box Optimization (BBO)
$\mathcal{X}^* = \arg\min_{x} f(x)$
s.t. $x \in \Omega \cap \Xi$
Unconstrained
$\Xi = \mathbb{R}^n$
Deterministic methods (RBF)
Gutmann-RBF
MSRS (Global and Local MSRBF)
CORS (CORS-RBF)
GLIS
GLIS-r
gMRS
Bayesian methods ($\mathcal{GP}$)
Probability of improvement
Expected improvement (EGO)
Lower confidence bound
gMRS
Constrained
$\Xi = \{x: g_\Xi(x) \le 0_{q_\Xi}\}$
$\Xi = \{x: u_\Xi(x) = 1\}$
Deterministic methods (RBF)
ConstrLMSRBF
COBRA
C-GLIS
C-GLIS-r
Bayesian methods ($\mathcal{GP}$)
Constrained Expected Improvement
SuperEGO
Constrained EGO with SVMs
Preference-Based Optimization (PBO)
$\mathcal{X}^* = \arg\min_{x} f(x)$
s.t. $x \in \Omega \cap \Xi$
$\equiv$
Find x^* such that $x^* \succsim x$
$\forall x \in \Omega \cap \Xi$
Unconstrained
$\Xi = \mathbb{R}^n$
Deterministic methods (RBF)
GLISp
GLISp-r
gMRS
Bayesian methods ($\mathcal{GP}$)
Probability of improvement
Expected improvement
Lower confidence bound
Dueling-Thompson sampling
gMRS
Constrained
$\Xi = \{x: g_\Xi(x) \le 0_{q_\Xi}\}$
$\Xi = \{x: u_\Xi(x) = 1\}$
Deterministic methods (RBF)
C-GLISp
C-GLISp-r
: Known analytical expressions
: Requires running computer simulations or performing real-world experiments
: Involves the interaction with a human decision-maker

Figure 9: Summary of the optimization frameworks and methods reviewed in Chapter 1 (global optimization), Chapter 2 (black-box optimization) and Chapter 3 (preference-based optimization). The algorithms proposed in this book are highlighted in magenta and will be covered in Chapter 5 and Chapter 6. gMRS [108] can be seen both as a deterministic and a Bayesian method since it can use any of the surrogates reviewed in Section 2.5 and Section 3.4.
]

Chapter 4. The GLIS, GLISp, C-GLIS and C-GLISp algorithms

This Chapter is devoted to reviewing four recent algorithms: GLobal minimum using Inverse distance weighting and Surrogate radial basis functions (GLIS [10]) for unconstrained BBO, GLISp [11] for unconstrained PBO, C-GLIS for constrained BBO, and C-GLISp [156] for constrained PBO. All methods use the radial basis function surrogate model $\hat{f}_N(\boldsymbol{x})$ in (2.20) to approximate either the black-box cost function or the latent scoring function of a human decision-maker. Compared to Bayesian and preferential Bayesian optimization methods, GLIS [10], GLISp [11], C-GLIS and C-GLISp [156] are more computationally efficient and show similar and, in some cases, better performances. *In the original articles, the matter of convergence to the global minimizer(s) of the global optimization problem in* (2.1) *is not addressed*. In Chapter 5, we present extensions of GLIS [10] and GLISp [11] that are globally convergent. We also revisit C-GLIS and C-GLISp [156] in Chapter 6, proposing a different surrogate for the black-box constraints functions and alternative infill sampling criteria.

The remainder of this Chapter is organized as follows. Section 4.1 gives a detailed review of the surrogate models used by GLIS [10], GLISp [11], C-GLIS and C-GLISp [156]. Then, Section 4.2 addresses the matter of exploration, which is particularly relevant for the infill sampling criteria of the four procedures, described in Section 4.3. Section 4.4 is devoted to examining additional algorithmic aspects, such as the rescaling of the decision variables and the recalibration of the shape parameter of $\hat{f}_N(\boldsymbol{x})$ in (2.20). Lastly, Section 4.5 provides the pseudocode for GLIS [10], GLISp [11], C-GLIS and C-GLISp [156], while Section 4.6 reports a brief summary of this Chapter.

4.1 Surrogate models

In this Section, we review the surrogate models used by algorithms GLIS [10], GLISp [11], C-GLIS and C-GLISp [156], which drive the search towards the global minimizer(s) of the GOP (2.1). The black-box cost function and the latent scoring function of the human-decision maker are approximated by the radial basis function expansion model in (2.20), which we report here:

$$\hat{f}_N\left(\boldsymbol{x}; \boldsymbol{\beta}_f, \epsilon_f\right) = \boldsymbol{\phi}_f\left(\boldsymbol{x}; \epsilon_f\right)^\top \cdot \boldsymbol{\beta}_f. \tag{4.1}$$

$\boldsymbol{\phi}_f\left(\boldsymbol{x}; \epsilon_f\right) \in \mathbb{R}^N$ in (4.1) is defined as in (2.21), i.e.:

$$\boldsymbol{\phi}_f\left(\boldsymbol{x}; \epsilon_f\right) = \begin{bmatrix} \varphi_f\left(\epsilon_f \cdot \|\boldsymbol{x} - \boldsymbol{x}_1\|_2\right) & \dots & \varphi_f\left(\epsilon_f \cdot \|\boldsymbol{x} - \boldsymbol{x}_N\|_2\right) \end{bmatrix}^\top.$$

Recall also the symmetric matrix in (2.26), $\Phi_f\left(\epsilon_f\right) \in \mathbb{R}^{N\times N}$, which originates from the samples in $\mathcal{X}$ (2.9):

$$\Phi_f\left(\epsilon_f\right) = \begin{bmatrix} \boldsymbol{\phi}_f\left(\boldsymbol{x}_1; \epsilon_f\right)^\top \\ \vdots \\ \boldsymbol{\phi}_f\left(\boldsymbol{x}_N; \epsilon_f\right)^\top \end{bmatrix}; \tag{4.2}$$

its (i, j)-th entry is $\Phi_f^{(i,j)}\left(\epsilon_f\right) = \varphi_f\left(\epsilon_f \cdot \left\|\boldsymbol{x}_i - \boldsymbol{x}_j\right\|_2\right)$. Matrix $\Phi_f\left(\epsilon_f\right)$ plays a key role in the computation of the weights $\boldsymbol{\beta}_f \in \mathbb{R}^N$ in (4.1), as we will review shortly in Section 4.1.1 and Section 4.1.2.

For what concerns `C-GLIS` and `C-GLISp` [156], the authors propose to use a surrogate model for the black-box constraints functions that is based on inverse distance weighting functions, as we will see in Section 4.1.3.

4.1.1 Surrogate model for the black-box cost function

In the context of black-box optimization, as we have seen in Section 2.5, the vector of weights $\boldsymbol{\beta}_f$ in (4.1) is obtained by enforcing the interpolation conditions:

$$\Phi_f\left(\epsilon_f\right) \cdot \boldsymbol{\beta}_f = \boldsymbol{y}, \tag{4.3}$$

where $\boldsymbol{y} \in \mathbb{R}^N$ are the measures of the cost function that are contained in $\mathcal{Y}$ (2.10). In practice, the linear system in (4.3) admits a unique solution if and only if matrix $\Phi_f\left(\epsilon_f\right)$ in (4.2) is non-singular. Some of the radial basis functions reported in Definition 2.6 guarantee that $\Phi_f\left(\epsilon_f\right)$ is positive definite (see Theorem 2.1 and Proposition 2.2). However, as claimed in [10], there are unavoidable numerical issues when the distances between the samples in $\mathcal{X}$ (2.9) get close to zero, which can easily happen as the surrogate-based optimization algorithm approaches the global minimizer(s) of the GOP (2.1). Furthermore, the shape parameter ϵ_f as well as the number and the distribution of the samples in $\mathcal{X}$ affect the condition number of $\Phi_f\left(\epsilon_f\right)$ [38, 119]. To compensate for these shortcomings, in [10] the author proposes to *solve the linear system in* (4.3) *using a low-rank approximation of* $\Phi_f\left(\epsilon_f\right)$ *in* (4.2) as follows:

1. Calculate the Singular Value Decomposition (SVD) [51] of $\Phi_f\left(\epsilon_f\right)$,

   $$\Phi_f\left(\epsilon_f\right) = U \cdot \Sigma \cdot V^\top,$$

   where $U, \Sigma, V \in \mathbb{R}^{N\times N}$. Σ is the diagonal matrix containing the singular values,

   $$\Sigma = \operatorname{diag}\left\{\sigma_1, \dots, \sigma_N\right\},$$

$\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_N \geq 0$, while U and V are two orthogonal matrices whose columns are the singular vectors;

2. Choose a threshold $\epsilon_{SVD} \in \mathbb{R}_{>0}$ and take a low-rank approximation of $\Phi_f\left(\epsilon_f\right)$ by:

    a) Removing all singular values lower than ϵ_{SVD}, obtaining the matrix:

    $$\hat{\Sigma} = \operatorname{diag}\left\{\sigma_1, \ldots, \sigma_{\hat{N}}\right\} \in \mathbb{R}^{\hat{N} \times \hat{N}},$$

    where $\hat{N} \leq N$ and $\sigma_1 \geq \ldots \geq \sigma_{\hat{N}} \geq \epsilon_{SVD}$;

    b) Extracting the first $\hat{N}$ columns of U and V, obtaining the matrices $\hat{U}, \hat{V} \in \mathbb{R}^{N \times \hat{N}}$;

    c) The low-rank approximation $\hat{\Phi}_f\left(\epsilon_f\right) \in \mathbb{R}^{N \times N}$ of $\Phi_f\left(\epsilon_f\right)$ in (4.2) is equal to:

    $$\hat{\Phi}_f\left(\epsilon_f\right) = \hat{U} \cdot \hat{\Sigma} \cdot \hat{V}^\top.$$

3. Finally, solve the linear system in (4.3) by taking advantage of the low-rank approximation $\hat{\Phi}_f\left(\epsilon_f\right)$ as follows:

$$\begin{aligned}
\boldsymbol{\beta}_f &= \Phi_f\left(\epsilon_f\right)^{-1} \cdot \boldsymbol{y} \\
&= \left(U \cdot \Sigma \cdot V^\top\right)^{-1} \cdot \boldsymbol{y} \\
&= \left(V^\top\right)^{-1} \cdot \Sigma^{-1} \cdot U^{-1} \cdot \boldsymbol{y} \\
&\quad \text{U and V are orthogonal matrices, i.e. } U^{-1} = U^\top \text{ and } V^{-1} = V^\top \\
&= V \cdot \Sigma^{-1} \cdot U^\top \cdot \boldsymbol{y} \\
&\quad \text{Substitute the low-rank approximation of } \Phi_f\left(\epsilon_f\right) \\
&\approx \hat{V} \cdot \hat{\Sigma}^{-1} \cdot \hat{U}^\top \cdot \boldsymbol{y}.
\end{aligned} \tag{4.4}$$

Note that, by construction, $\hat{\Sigma}$ is always invertible.

When solving the linear system in (4.3) through a low-rank approximation of $\Phi_f\left(\epsilon_f\right)$ in (4.2), an issue might arise: $\hat{f}_N\left(\boldsymbol{x}; \boldsymbol{\beta}_f, \epsilon_f\right)$ *in* (4.1) *might not interpolate all the points in*

$$\mathcal{D} = \left\{(\boldsymbol{x}_i, y_i) : \boldsymbol{x}_i \in \mathcal{X}, y_i \in \mathcal{Y}, i = 1, \ldots, N\right\}.$$

Moreover, the shape parameter ϵ_f, as well as the choice of the radial function $\varphi_f(\cdot)$, influence the quality of the approximation provided by the surrogate model and the successful interpolation of the points in $\mathcal{D}$. However, by taking a low rank approximation of (4.2), we ensure the existence of a unique $\boldsymbol{\beta}_f$ and avoid numerical problems linked to a possibly high condition number for $\Phi_f\left(\epsilon_f\right)$. Also, as

noted in [10], *this approach can be particularly useful when the measures of $f(\boldsymbol{x})$ are affected by noise*.
The following Example shows that computing $\boldsymbol{\beta}_f$ as in (4.4) can lead to a non-interpolating $\hat{f}_N\left(\boldsymbol{x}; \boldsymbol{\beta}_f, \epsilon_f\right)$ in (4.1).

Example 4.1: Interpolating and non-interpolating surrogates

Consider the `gramacy and lee` [53] benchmark function defined as:

$$f(x) = \frac{\sin\left(10 \cdot \pi \cdot x^{(1)}\right)}{2 \cdot x^{(1)}} + \left(x^{(1)} - 1\right)^4.$$

We generate a set $\mathcal{X}$ in (2.9) of $N = 20$ samples, using a latin hypercube design (see Section 2.4), and measure $f(x)$ at those points. Then, we fit four surrogates $\hat{f}_N\left(x; \boldsymbol{\beta}_f, \epsilon_f\right)$ following the aforementioned approach, with $\epsilon_{SVD} = 10^{-6}$ and with the following hyper-parameters:

1. $\varphi_f(\cdot)$ inverse quadratic and $\epsilon_f = 1$,
2. $\varphi_f(\cdot)$ inverse quadratic and $\epsilon_f = 10$,
3. $\varphi_f(\cdot)$ thin plate spline and $\epsilon_f = 1$,
4. $\varphi_f(\cdot)$ linear and $\epsilon_f = 1$.

Figure 10 compares $f(x)$ with $\hat{f}_N\left(x; \boldsymbol{\beta}_f, \epsilon_f\right)$ in the four cases. Notice how the latter three models interpolate all the points in $\mathcal{D}$ whereas the first one does not.

We conclude this Section with another Example that shows the performances of the surrogate model in (4.1), with $\boldsymbol{\beta}_f$ computed as in (4.4), when used to approximate two-dimensional cost functions.

Example 4.2: Examples of two-dimensional surrogates

Consider the `camel six humps` [62] and the `adjiman` [62] benchmark functions, defined respectively as:

$$\begin{aligned} f(\boldsymbol{x}) = &\left[4 - 2.1 \cdot \left(x^{(1)}\right)^2 + \frac{\left(x^{(1)}\right)^4}{3}\right] \cdot \left(x^{(1)}\right)^2 + x^{(1)} \cdot x^{(2)} + \\ &+ \left[4 \cdot \left(x^{(2)}\right)^2 - 4\right] \cdot \left(x^{(2)}\right)^2 \text{ and} \\ f(\boldsymbol{x}) = &\cos\left(x^{(1)}\right) \cdot \sin\left(x^{(2)}\right) - \frac{x^{(1)}}{\left(x^{(2)}\right)^2 + 1}. \end{aligned}$$

We generate a set $\mathcal{X}$ in (2.9) of $N = 100$ samples, using a latin hypercube design (see Section

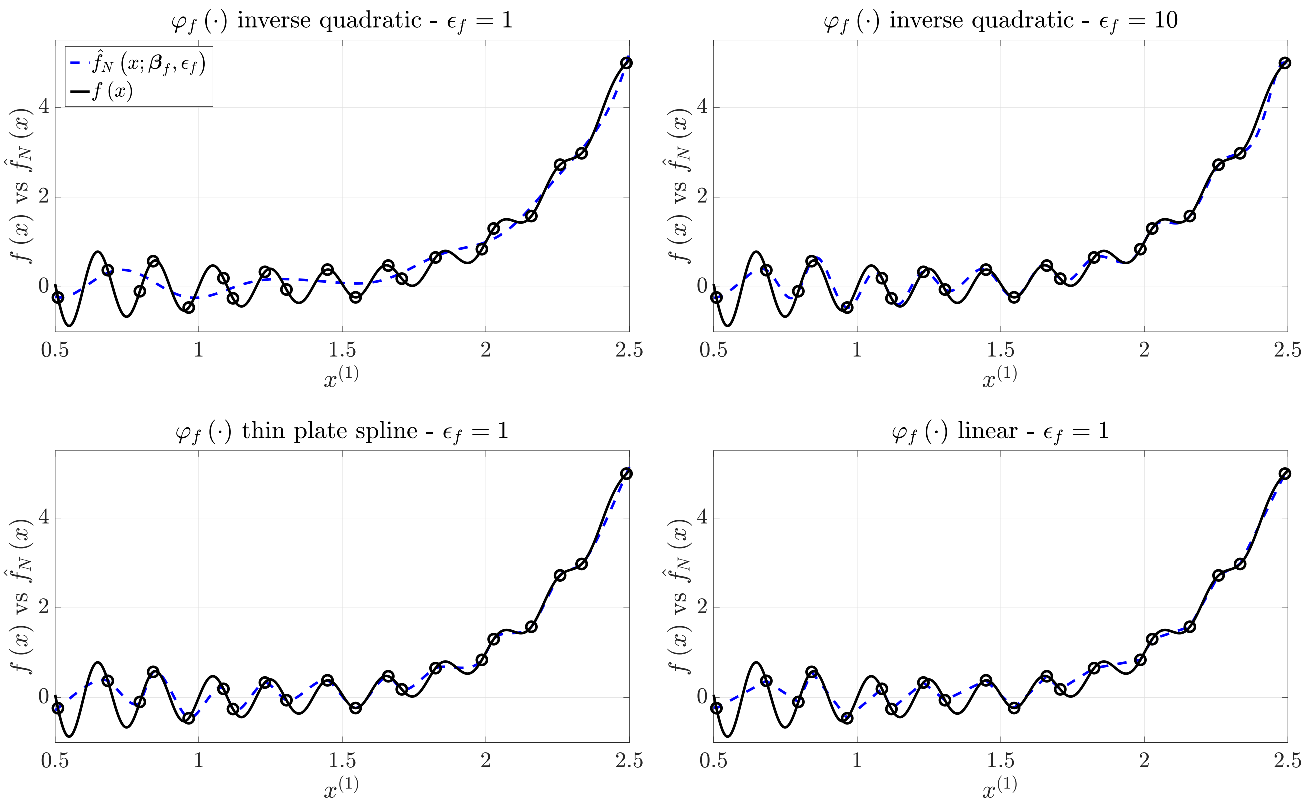


Figure 10: Comparison between the real function $f(x)$ (black line), which is the one-dimensional `gramacy and lee` [53] function, and the surrogate model $\hat{f}_N\left(x;\boldsymbol{\beta}_f,\epsilon_f\right)$ (dashed blue line), estimated as described in Example 4.1.

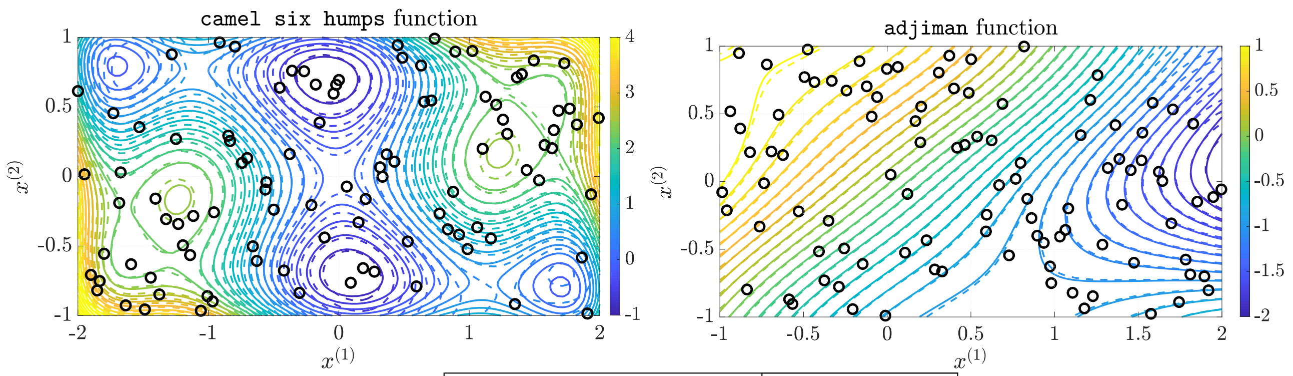
2.4), and measure $f(\boldsymbol{x})$ at those points. Then, for each $f(\boldsymbol{x})$, we fit a surrogate as described in this Section, with $\epsilon_{SVD} = 10^{-6}$, $\varphi_f(\cdot)$ inverse quadratic and $\epsilon_f = 1$. Figure 11 compares $f(\boldsymbol{x})$ with $\hat{f}_N\left(x;\boldsymbol{\beta}_f,\epsilon_f\right)$ and reports the *Root Mean Square Error (RMSE)*,

$$RMSE = \sqrt{\frac{1}{N}\cdot\sum_{i=1}^{N}\left[\hat{f}_N\left(\boldsymbol{x}_i;\boldsymbol{\beta}_f,\epsilon_f\right)-y_i\right]^2},$$

obtained by the approximations. Notice that the RMSEs are not exactly zero, which means that the surrogates do not interpolate all the points in $\mathcal{D}$. However, in both cases, $\hat{f}_N\left(x;\boldsymbol{\beta}_f,\epsilon_f\right)$ capture the shape of $f(\boldsymbol{x})$ quite well.

4.1.2 Surrogate model for the preference function and the scoring function

In the preference-based optimization framework, there are no interpolation conditions to be enforced. Instead, in `GLISp` [11], the weight vector $\boldsymbol{\beta}_f$ in (4.1) is selected so that the surrogate preference



	camel six humps [62]	adjiman [62]
RMSE	$12.2 \cdot 10^{-5}$	$1.71 \cdot 10^{-5}$

Figure 11: Comparison between the real function $f(x)$ (continuous line), respectively the `camel six humps` [62] function on the left and the `adjiman` [62] function on the right, and the surrogate model $\hat{f}_N\left(x; \beta_f, \epsilon_f\right)$ (dashed line), estimated as described in Example 4.2. We also report the RMSEs of both approximations.

function $\hat{\pi}_{\gtrsim_N}(\boldsymbol{x}_i, \boldsymbol{x}_j; \boldsymbol{\beta}_f, \epsilon_f)$ in (3.11), namely:

$$\hat{\pi}_{\gtrsim_N}(\boldsymbol{x}_i, \boldsymbol{x}_j; \boldsymbol{\beta}_f, \epsilon_f) = \begin{cases} -1 & \text{if } \hat{f}_N\left(\boldsymbol{x}_i; \boldsymbol{\beta}_f, \epsilon_f\right) - \hat{f}_N\left(\boldsymbol{x}_j; \boldsymbol{\beta}_f, \epsilon_f\right) \leq -\sigma_\pi \\ 0 & \text{if } \left|\hat{f}_N\left(\boldsymbol{x}_i; \boldsymbol{\beta}_f, \epsilon_f\right) - \hat{f}_N\left(\boldsymbol{x}_j; \boldsymbol{\beta}_f, \epsilon_f\right)\right| \leq \sigma_\pi \\ 1 & \text{if } \hat{f}_N\left(\boldsymbol{x}_i; \boldsymbol{\beta}_f, \epsilon_f\right) - \hat{f}_N\left(\boldsymbol{x}_j; \boldsymbol{\beta}_f, \epsilon_f\right) \geq \sigma_\pi \end{cases}, \tag{4.5}$$

correctly describes the preferences inside the sets $\mathcal{B}$ (3.9) and $\mathcal{S}$ (3.10), resulting in the following optimization problem (see Section 3.4.1):

$$\begin{aligned} \arg \min_{\boldsymbol{\varepsilon}_f, \boldsymbol{\beta}_f} \quad & \frac{\lambda_f}{2} \cdot \boldsymbol{\beta}_f^\top \cdot \boldsymbol{\beta}_f + \boldsymbol{r}_f^\top \cdot \boldsymbol{\varepsilon}_f & \quad (4.6) \\ \text{s.t.} \quad & \hat{f}_N\left(\boldsymbol{x}_{\ell(h)}; \boldsymbol{\beta}_f, \epsilon_f\right) - \hat{f}_N\left(\boldsymbol{x}_{\kappa(h)}; \boldsymbol{\beta}_f, \epsilon_f\right) \leq -\sigma_\pi + \varepsilon_f^{(h)} & \forall h : b_h = -1 \\ & \left|\hat{f}_N\left(\boldsymbol{x}_{\ell(h)}; \boldsymbol{\beta}_f, \epsilon_f\right) - \hat{f}_N\left(\boldsymbol{x}_{\kappa(h)}; \boldsymbol{\beta}_f, \epsilon_f\right)\right| \leq \sigma_\pi + \varepsilon_f^{(h)} & \forall h : b_h = 0 \\ & \hat{f}_N\left(\boldsymbol{x}_{\ell(h)}; \boldsymbol{\beta}_f, \epsilon_f\right) - \hat{f}_N\left(\boldsymbol{x}_{\kappa(h)}; \boldsymbol{\beta}_f, \epsilon_f\right) \geq \sigma_\pi - \varepsilon_f^{(h)} & \forall h : b_h = 1 \\ & \varepsilon_f^{(h)} \geq 0 \\ & h = 1, \dots, M. \end{aligned}$$

Recall that *the goal of preference-based optimization is not to approximate the scoring function of the human decision-maker, $f(\boldsymbol{x})$, in the best way possible but to converge to a global minimizer of the GOP* (2.1) *with the least amount of queries to the DM*. Therefore, the preferences related to the best-found candidate at a certain iteration of the PBO procedure, namely $\boldsymbol{x}_{best}(N)$, are more relevant than the others, making us interested in satisfying them without any slack. In order to do so, in GLISp

[11], vector $\boldsymbol{r}_f$ is defined as:

$$r_f^{(h)} = 1, \quad \forall h : (\ell(h), \kappa(h)) \in \mathcal{S}, \boldsymbol{x}_{\ell(h)} \neq \boldsymbol{x}_{\boldsymbol{best}}(N) \text{ and } \boldsymbol{x}_{\kappa(h)} \neq \boldsymbol{x}_{\boldsymbol{best}}(N), \tag{4.7a}$$

$$r_f^{(h)} = 10, \quad \forall h : (\ell(h), \kappa(h)) \in \mathcal{S}, \boldsymbol{x}_{\ell(h)} = \boldsymbol{x}_{\boldsymbol{best}}(N) \text{ or } \boldsymbol{x}_{\kappa(h)} = \boldsymbol{x}_{\boldsymbol{best}}(N), \tag{4.7b}$$

penalizing more the slacks associated to the preferences expressed on the current best candidate.

Now, we show that Problem (4.6) is a Quadratic Program (QP) for $\lambda_f \in \mathbb{R}_{>0}$ and a Linear Program (LP) for $\lambda_f = 0$. Consider the terms:

$$\hat{f}_N\left(\boldsymbol{x}_{\ell(h)}; \boldsymbol{\beta}_f, \epsilon_f\right) - \hat{f}_N\left(\boldsymbol{x}_{\kappa(h)}; \boldsymbol{\beta}_f, \epsilon_f\right),$$

that are on the left side of the inequalities that concern the preferences in Problem (4.6). By substituting (4.1), we have that:

$$\begin{aligned}\hat{f}_N\left(\boldsymbol{x}_{\ell(h)}; \boldsymbol{\beta}_f, \epsilon_f\right) - \hat{f}_N\left(\boldsymbol{x}_{\kappa(h)}; \boldsymbol{\beta}_f, \epsilon_f\right) &= \boldsymbol{\phi}_f\left(\boldsymbol{x}_{\ell(h)}; \epsilon_f\right)^\top \cdot \boldsymbol{\beta}_f - \boldsymbol{\phi}_f\left(\boldsymbol{x}_{\kappa(h)}; \epsilon_f\right)^\top \cdot \boldsymbol{\beta}_f \\ &= \left[\boldsymbol{\phi}_f\left(\boldsymbol{x}_{\ell(h)}; \epsilon_f\right) - \boldsymbol{\phi}_f\left(\boldsymbol{x}_{\kappa(h)}; \epsilon_f\right)\right]^\top \cdot \boldsymbol{\beta}_f. \end{aligned} \tag{4.8}$$

Let us consider the preferences belonging to a certain category, for example all the preferences in $\mathcal{B}$ (3.9) which are such that $b_h = -1$. Denote as $M_{-1} \in \mathbb{N} \cup \{0\}, M_{-1} \leq M, M = |\mathcal{B}|$, the number of preferences expressed by the DM that pertain to that category. We define the matrix $\Phi_{f_{-1}}\left(\epsilon_f\right) \in \mathbb{R}^{M_{-1} \times N}$ as:

$$\Phi_{f_{-1}}\left(\epsilon_f\right) = \begin{bmatrix} \left\{\boldsymbol{\phi}_f\left(\boldsymbol{x}_{\ell(1)}; \epsilon_f\right) - \boldsymbol{\phi}_f\left(\boldsymbol{x}_{\kappa(1)}; \epsilon_f\right)\right\}^\top \\ \vdots \\ \left\{\boldsymbol{\phi}_f\left(\boldsymbol{x}_{\ell(M_{-1})}; \epsilon_f\right) - \boldsymbol{\phi}_f\left(\boldsymbol{x}_{\kappa(M_{-1})}; \epsilon_f\right)\right\}^\top \end{bmatrix}, \tag{4.9}$$

whose (h, j)-th entry is:

$$\Phi_{f_{-1}}^{(h,j)}\left(\epsilon_f\right) = \varphi_f\left(\epsilon_f \cdot \left\|\boldsymbol{x}_{\ell(h)} - \boldsymbol{x}_j\right\|_2\right) - \varphi_f\left(\epsilon_f \cdot \left\|\boldsymbol{x}_{\kappa(h)} - \boldsymbol{x}_j\right\|_2\right).$$

In (4.9), for the sake of simplicity and without loss of generality, we have assumed that $b_h = -1$ for $h = 1, \ldots, M_{-1}$ in $\mathcal{B}$ (3.9), although it need not be the case. In practice, matrix $\Phi_{f_{-1}}\left(\epsilon_f\right)$ is built by subtracting the rows of matrix $\Phi_f\left(\epsilon_f\right)$ in (4.2) indexed by those elements of $\mathcal{S}$ (3.10) that are associated to the preferences in $\mathcal{B}$ (3.9) which are such that $b_h = -1$. We can define matrices $\Phi_{f_0}\left(\epsilon_f\right) \in \mathbb{R}^{M_0 \times N}$ and $\Phi_{f_1}\left(\epsilon_f\right) \in \mathbb{R}^{M_1 \times N}$, $M_0, M_1 \in \mathbb{N} \cup \{0\}$, associated to the other two categories of preferences, in the same fashion. Similarly, we can split the vector of slack variables $\boldsymbol{\varepsilon}_f \in \mathbb{R}^M_{\geq 0}$ into three separate vectors, $\boldsymbol{\varepsilon}_{f_{-1}} \in \mathbb{R}^{M_{-1}}_{\geq 0}, \boldsymbol{\varepsilon}_{f_0} \in \mathbb{R}^{M_0}_{\geq 0}$ and $\boldsymbol{\varepsilon}_{f_1} \in \mathbb{R}^{M_1}_{\geq 0}$, which correspond to the different types of preferences. Now, using (4.8) and (4.9), we can re-write Problem (4.6) as:

$$\begin{aligned}
&\arg\min_{\boldsymbol{\varepsilon}_f, \boldsymbol{\beta}_f} \frac{\lambda_f}{2} \cdot \boldsymbol{\beta}_f^\top \cdot \boldsymbol{\beta}_f + \boldsymbol{r}_f^\top \cdot \boldsymbol{\varepsilon}_f \\
\text{s.t.}\quad & \boldsymbol{\Phi}_{f_{-1}}\left(\epsilon_f\right) \cdot \boldsymbol{\beta}_f - \boldsymbol{\varepsilon}_{f_{-1}} \leq -\sigma_\pi \cdot \mathbf{1}_{M_{-1}} \\
& \boldsymbol{\Phi}_{f_0}\left(\epsilon_f\right) \cdot \boldsymbol{\beta}_f - \boldsymbol{\varepsilon}_{f_0} \leq \sigma_\pi \cdot \mathbf{1}_{M_0} \\
& \boldsymbol{\Phi}_{f_0}\left(\epsilon_f\right) \cdot \boldsymbol{\beta}_f + \boldsymbol{\varepsilon}_{f_0} \geq -\sigma_\pi \cdot \mathbf{1}_{M_0} \\
& \boldsymbol{\Phi}_{f_1}\left(\epsilon_f\right) \cdot \boldsymbol{\beta}_f + \boldsymbol{\varepsilon}_{f_1} \geq \sigma_\pi \cdot \mathbf{1}_{M_1} \\
& \boldsymbol{\varepsilon}_f \geq \mathbf{0}_M.
\end{aligned}$$

Finally, by considering the augmented vector $\boldsymbol{\xi}_f = \begin{bmatrix} \boldsymbol{\beta}_f^\top & \boldsymbol{\varepsilon}_f^\top \end{bmatrix}^\top \in \mathbb{R}^{(N+M)}$, we can formulate Problem (4.6) in standard QP form:

$$\begin{aligned}
&\arg\min_{\boldsymbol{\xi}_f} \frac{1}{2} \cdot \boldsymbol{\xi}_f^\top \cdot H_{QP} \cdot \boldsymbol{\xi}_f + \boldsymbol{r_{QP}}^\top \cdot \boldsymbol{\xi}_f \\
\text{s.t.}\quad & A_{QP} \cdot \boldsymbol{\xi}_f \leq \boldsymbol{b_{QP}},
\end{aligned} \tag{4.10}$$

where:

$$\begin{aligned}
H_{QP} &= \begin{bmatrix} \lambda_f \cdot I_{N\times N} & 0_{N\times M} \\ 0_{M\times N} & 0_{M\times M} \end{bmatrix} \in \mathbb{R}^{(N+M)\times(N+M)}, \\
\boldsymbol{r_{QP}} &= \begin{bmatrix} \mathbf{0}_N \\ \boldsymbol{r}_f \end{bmatrix} \in \mathbb{R}^{(N+M)}, \\
A_{QP} &= \begin{bmatrix} \boldsymbol{\Phi}_{f_{-1}}\left(\epsilon_f\right) & -I_{M_{-1}\times M_{-1}} & 0_{M_{-1}\times M_0} & 0_{M_{-1}\times M_1} \\ \boldsymbol{\Phi}_{f_0}\left(\epsilon_f\right) & 0_{M_0\times M_{-1}} & -I_{M_0\times M_0} & 0_{M_0\times M_1} \\ -\boldsymbol{\Phi}_{f_0}\left(\epsilon_f\right) & 0_{M_0\times M_{-1}} & -I_{M_0\times M_0} & 0_{M_0\times M_1} \\ -\boldsymbol{\Phi}_{f_1}\left(\epsilon_f\right) & 0_{M_1\times M_{-1}} & 0_{M_1\times M_0} & -I_{M_1\times M_1} \\ 0_{M\times N} & \multicolumn{3}{c}{-I_{M\times M}} \end{bmatrix} \in \mathbb{R}^{(2\cdot M+M_0)\times(N+M)}, \\
\boldsymbol{b_{QP}} &= \begin{bmatrix} -\sigma_\pi \cdot \mathbf{1}_{M_{-1}} \\ \sigma_\pi \cdot \mathbf{1}_{M_0} \\ \sigma_\pi \cdot \mathbf{1}_{M_0} \\ -\sigma_\pi \cdot \mathbf{1}_{M_1} \\ \mathbf{0}_M \end{bmatrix} \in \mathbb{R}^{(2\cdot M+M_0)}.
\end{aligned}$$

We can easily prove the following Proposition concerning Problem (4.10).

Proposition 4.1. *Problem* (4.10) *is convex for any* $\lambda_f \in \mathbb{R}_{\geq 0}$.

Proof. In the case $\lambda_f = 0$, Problem (4.10) is a LP and thus convex by construction [18]. Instead, for $\lambda_f \in \mathbb{R}_{>0}$, Problem (4.10) is a QP, hence we need to check if H_{QP} is positive semidefinite [18]. In this case, H_{QP} is a diagonal matrix,

$$H_{QP} = \text{diag}\left\{\lambda_f, \ldots, \lambda_f, 0, \ldots, 0\right\},$$

with N positive eigenvalues and M zero-valued eigenvalues. Therefore, H_{QP} is positive semidefinite [51] and Problem (4.10) is convex. □

To conclude this Section, we show two examples of surrogate models in (4.1) with $\boldsymbol{\beta}_f$ computed by solving Problem (4.10).

Example 4.3: Preference ordering

Consider the following preference ordering:

$$x_3 \succ x_1 \succ x_2,$$

which results in a scoring function such that:

$$f(x_3) < f(x_1) < f(x_2).$$

Suppose that:

$$\mathcal{X} = \{x_1 = 1, x_2 = 4, x_3 = 3\}$$

and consider the set of preferences and the mapping set originated from the previous ordering:

$$\mathcal{B} = \{b_1 = -1, b_2 = 1, b_3 = 1\},$$
$$\mathcal{S} = \{(1,2), (2,3), (1,3)\}.$$

We estimate four surrogate models with:

1. $\varphi_f(\cdot)$ inverse quadratic and $\epsilon_f = 0.1$,
2. $\varphi_f(\cdot)$ inverse quadratic and $\epsilon_f = 1$,
3. $\varphi_f(\cdot)$ inverse quadratic and $\epsilon_f = 10$,
4. $\varphi_f(\cdot)$ linear and $\epsilon_f = 1$.

The hyper-parameters for Problem (4.6) are: $\lambda_f = 10^{-6}, \sigma_\pi = 1$ and $\boldsymbol{r}_f = \mathbf{1}_3$. Figure 12 depicts the obtained surrogate models. Notice how all $\hat{f}_N\left(x; \boldsymbol{\beta}_f, \epsilon_f\right)$ capture the order imposed by the preferences. Furthermore, increasing the value of the shape parameter ϵ_f results in surrogates which show "more local" behaviors (cf. Figure 6).

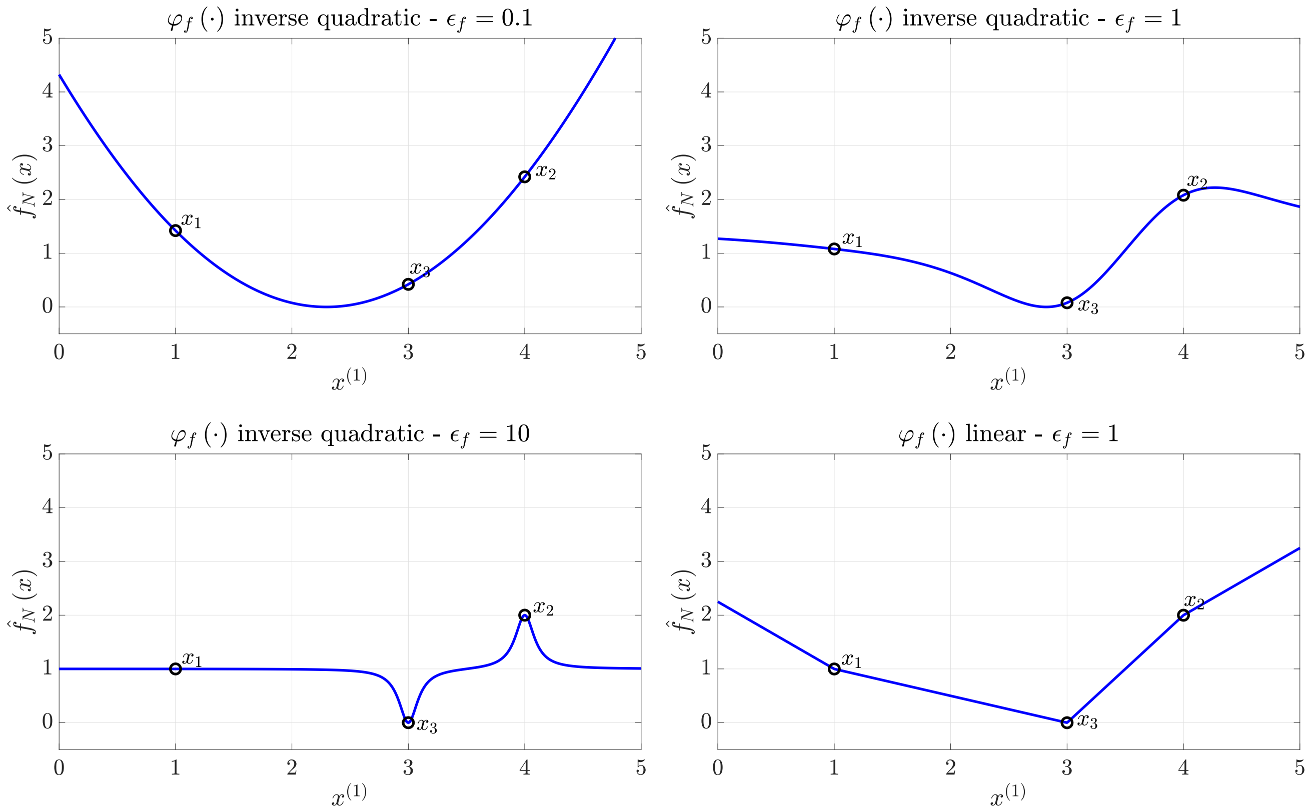


Figure 12: Surrogate models obtained as described in Example 4.3. For ease of comparison, all the surrogates have been shifted so that their minimum values are zero.

Example 4.4: Examples of two-dimensional surrogates

Consider the `camel six humps` [62] and the `adjiman` [62] benchmark functions defined as in Example 4.2. We use the same sets of samples $\mathcal{X}$ (2.9) generated in Example 4.2 to estimate the surrogate model in (4.1) but in the preference-based setting. To that end, we express $M = N - 1$ preferences following Algorithm 7, which exploits the transitive property of preference relations (see Section 3.1) to keep track of the best candidate. Then, we compute the vector of weights $\boldsymbol{\beta}_f$ by solving Problem (4.10) and with hyper-parameters: $\varphi_f(\cdot)$ inverse quadratic, $\epsilon_f = 1$, $\lambda_f = 10^{-6}, \sigma_\pi = 10^{-2}$ and $\boldsymbol{r}_f = \mathbf{1}_M$. Figure 13 compares the obtained surrogate models with the real scoring functions. Clearly, $\hat{f}_N\left(\boldsymbol{x}; \boldsymbol{\beta}_f, \epsilon_f\right)$ in the PBO setting are not as accurate as their BBO counterparts (cf. Figure 11). However, at least roughly, they capture the most promising regions of $f(\boldsymbol{x})$.

4.1.3 Surrogate model for the black-box constraints functions

`C-GLIS` and `C-GLISp` [156] rely on inverse distance weighting interpolation [10] to estimate the probability of a sample being Ξ-feasible. We start by defining what inverse distance weighting functions are.

Algorithm 7: Initial queries for preference-based optimization

Input: (i) Initial set of samples $\mathcal{X}$, $|\mathcal{X}| = N_{init}$, $N_{init} \geq 2$, in (2.9).

Output: (i) Set of preferences $\mathcal{B}$ in (3.9); (ii) Mapping set $\mathcal{S}$ in (3.10); (iii) Initial best sample $\boldsymbol{x_{best}}\left(N_{init}\right)$.

```
1: Initialize the best candidate as $\boldsymbol{x_{best}}(1) = \boldsymbol{x}_1$, $i_{best} = 1$
2: Initialize the sets $\mathcal{B}$ and $\mathcal{S}$: $\mathcal{B} = \emptyset$ and $\mathcal{S} = \emptyset$
3: for $i = 2$ to $|\mathcal{X}| = N_{init}$ do
4:     Let the human decision-maker express a preference between $\boldsymbol{x_{best}}(i-1)$ and $\boldsymbol{x}_i$, obtaining $b = \pi_{\gtrsim}\left(\boldsymbol{x_{best}}(i-1), \boldsymbol{x}_i\right)$
5:     Update the sets $\mathcal{B}$ and $\mathcal{S}$: $\mathcal{B} = \mathcal{B} \cup \{b\}$ and $\mathcal{S} = \mathcal{S} \cup \{(i_{best}, i)\}$
6:     if $b = 1$ (i.e. $\boldsymbol{x}_i \succ \boldsymbol{x_{best}}(i-1)$) then
7:         Update the best candidate, $\boldsymbol{x_{best}}(i) = \boldsymbol{x}_i$ and $i_{best} = i$
8:     else
9:         Keep the best candidate unaltered, $\boldsymbol{x_{best}}(i) = \boldsymbol{x_{best}}(i-1)$
```

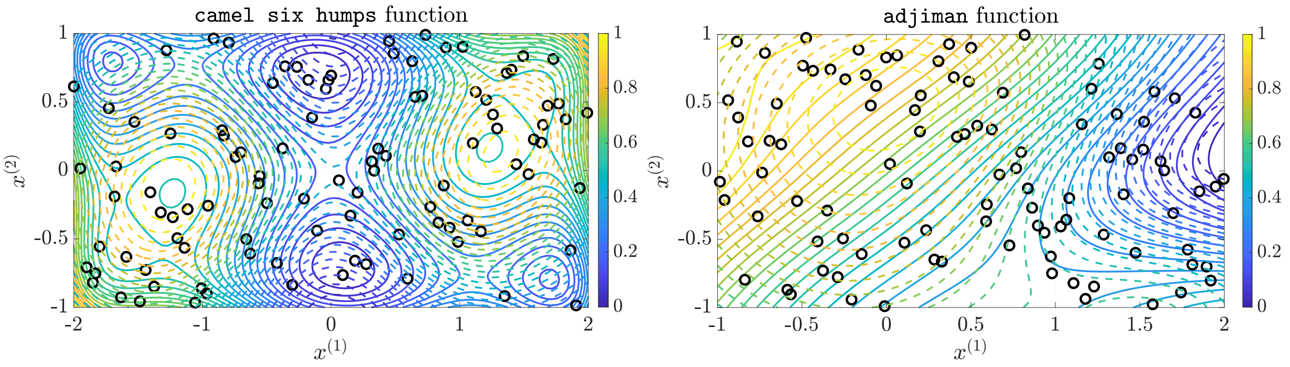


Figure 13: Comparison between the real function $f(\boldsymbol{x})$ (continuous line), respectively the `camel six humps` [62] function on the left and the `adjiman` [62] function on the right, and the surrogate model $\hat{f}_N\left(\boldsymbol{x}; \boldsymbol{\beta}_f, \epsilon_f\right)$ (dashed line), estimated as described in Example 4.4. The surrogate models and the scoring functions are made comparable by rescaling them to the $[0, 1]$ range.

Definition 4.1: Inverse distance weighting functions [132]. *Given a sample $\boldsymbol{x}_i \in \mathcal{X}$ in (2.9), we define its corresponding Inverse Distance Weighting (IDW) function $w_i : \mathbb{R}^n \setminus \{\boldsymbol{x}_i\} \to \mathbb{R}_{>0}$ as:*

$$w_i(\boldsymbol{x}) = \frac{1}{\|\boldsymbol{x} - \boldsymbol{x}_i\|_2^2}. \tag{4.11}$$

We also define an alternative weighting function as [69]:

$$\tilde{w}_i(\boldsymbol{x}) = \frac{\exp\left\{-\|\boldsymbol{x} - \boldsymbol{x}_i\|_2^2\right\}}{\|\boldsymbol{x} - \boldsymbol{x}_i\|_2^2}. \tag{4.12}$$

Compared to $w_i(\boldsymbol{x})$ in (4.11), $\tilde{w}_i(\boldsymbol{x})$ in (4.12) tends to zero faster for $\|\boldsymbol{x} - \boldsymbol{x}_i\|_2^2 \to \infty$, see Figure 14.

We can easily prove the following Proposition concerning inverse distance weighting functions.

Proposition 4.2: Differentiability of IDW functions. *Both $w_i(\boldsymbol{x})$ in (4.11) and $\tilde{w}_i(\boldsymbol{x})$ in (4.12) are differentiable on $\mathbb{R}^n \setminus \{\boldsymbol{x}_i\}$.*

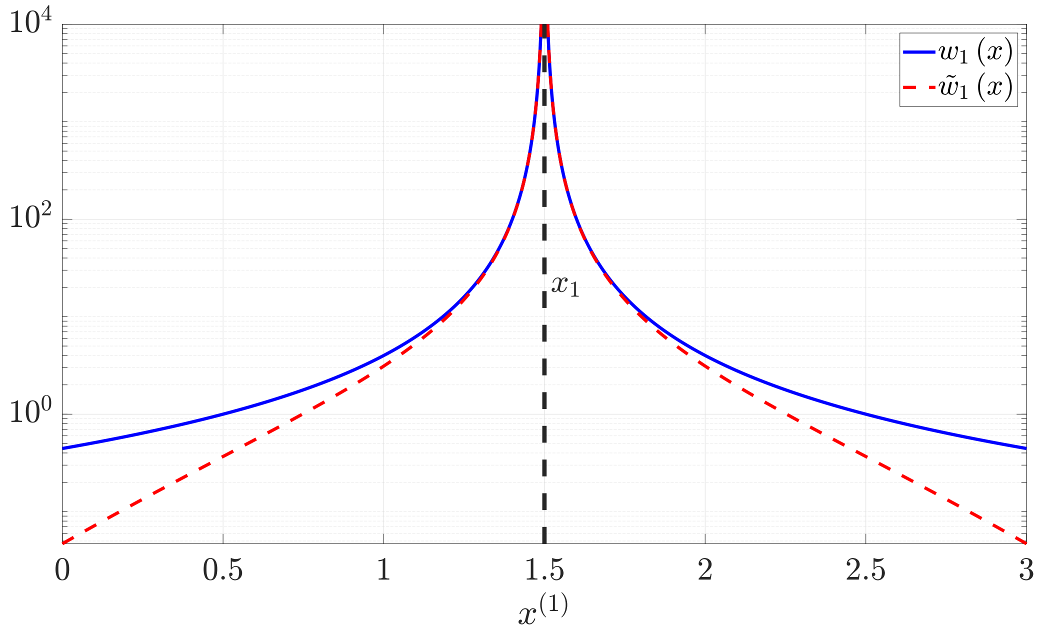


Figure 14: One-dimensional comparison between the two different definitions of the IDW functions, namely $w_i(\boldsymbol{x})$ in (4.11) (blue line) and $\tilde{w}_i(\boldsymbol{x})$ in (4.12) (dashed red line), defined starting from a sample $x_1 = 1.5$.

Proof. We know that the squared Euclidean norm $\|\boldsymbol{x} - \boldsymbol{x}_i\|_2^2$ is differentiable $\forall \boldsymbol{x} \in \mathbb{R}^n$ (see Example 2.4). Furthermore, we have that:

$$\|\boldsymbol{x} - \boldsymbol{x}_i\|_2^2 \neq 0, \quad \forall \boldsymbol{x} \in \mathbb{R}^n \setminus \{\boldsymbol{x}_i\}.$$

Therefore, due to the reciprocal rule, $w_i(\boldsymbol{x})$ in (4.11) is differentiable $\forall \boldsymbol{x} \in \mathbb{R}^n \setminus \{\boldsymbol{x}_i\}$. We can reach the same conclusion for $\tilde{w}_i(\boldsymbol{x})$ in (4.12), since it is the composition of functions that are differentiable $\forall \boldsymbol{x} \in \mathbb{R}^n \setminus \{\boldsymbol{x}_i\}$. □

Now, consider a generic multivariable function $h : \mathbb{R}^n \to \mathbb{R}$ and suppose that we measure its values at the samples in $\mathcal{X}$ (2.9), obtaining:

$$\mathcal{H} = \{h_i : h_i = h(\boldsymbol{x}_i), \boldsymbol{x}_i \in \mathcal{X}\}. \tag{4.13}$$

We define the inverse distance weighting interpolation function as follows.

Definition 4.2: Inverse distance weighting interpolation function [69]. *Consider the set of samples $\mathcal{X}$ in (2.9) and define the weighting functions $v_i : \mathbb{R}^n \to \mathbb{R}$ as:*

$$v_i(\boldsymbol{x}) = \begin{cases} 1 & \text{if } \boldsymbol{x} = \boldsymbol{x}_i \\ 0 & \text{if } \boldsymbol{x} = \boldsymbol{x}_j, j \neq i \\ \frac{\tilde{w}_i(\boldsymbol{x})}{\sum_{j=1}^{N} \tilde{w}_j(\boldsymbol{x})} & \text{otherwise} \end{cases}. \tag{4.14}$$

The Inverse Distance Weighting Interpolation (IDWI) function, $\hat{h}_N : \mathbb{R}^n \to \mathbb{R}$, is defined as the linear combination of the $v_i(\boldsymbol{x})$'s in (4.14) *with the measures of $h(\boldsymbol{x})$ in* (4.13)*, namely:*

$$\hat{h}_N(\boldsymbol{x}) = \sum_{i=1}^{N} v_i(\boldsymbol{x}) \cdot h_i. \tag{4.15}$$

The next Proposition highlights the properties of the IDWI function in (4.15).

Proposition 4.3: Properties of the IDWI function [10]. *The inverse distance weighting interpolation function $\hat{h}_N(\boldsymbol{x})$ in* (4.15) *exhibits the following properties:*

1. $\hat{h}_N(\boldsymbol{x}_i) = h_i, \forall i = 1, \ldots, N$ *(interpolation),*
2. $\min_{h_i \in \mathcal{H}} h_i \leq \hat{h}_N(\boldsymbol{x}) \leq \max_{h_i \in \mathcal{H}} h_i, \forall \boldsymbol{x} \in \mathbb{R}^n$ *(bounded range),*
3. $\hat{h}_N(\boldsymbol{x})$ *is differentiable everywhere and* $\nabla_{\boldsymbol{x}} \hat{h}_N(\boldsymbol{x}_i) = \mathbf{0}_n, \forall i = 1, \ldots, N$.

Now, consider the Ξ-feasibility information contained inside the set $\mathcal{U}_\Xi$ in (2.11). Proposition 4.3 highlights how the IDWI function in (4.15) can be used as a surrogate for the probability of Ξ-feasibility as follows:

$$\begin{aligned} p_N(\boldsymbol{x} \in \Xi) &= p\left(\boldsymbol{x} \in \Xi \,\middle|\, \mathcal{U}_\Xi, \mathcal{X}, \boldsymbol{x}\right) \\ &= \sum_{i=1}^{N} v_i(\boldsymbol{x}) \cdot u_i. \end{aligned} \tag{4.16}$$

From Property 1 of Proposition 4.3, the IDWI function associates to samples $\boldsymbol{x}_i \in \mathcal{X}$ that are Ξ-feasible ($u_i = 1$) a probability $p_N(\boldsymbol{x} \in \Xi) = 1$ (and vice-versa for the Ξ-infeasible ones). Furthermore, Property 2 of Proposition 4.3 ensures that $p_N(\boldsymbol{x} \in \Xi)$ in (4.16) has range $[0, 1]$, as it should be for any probability.

We conclude this Section with some examples on the surrogate probability of Ξ-feasibility in (4.16).

Example 4.5: Examples of two-dimensional surrogates

Consider the following Ξ-feasible regions defined for $\boldsymbol{x} \in \mathbb{R}^2$:

1. $\Xi_1 = \left\{\boldsymbol{x} : \begin{bmatrix} -0.25 & -0.25 \end{bmatrix}^\top \leq \boldsymbol{x} \leq \begin{bmatrix} 0.9 & 0.4 \end{bmatrix}^\top \right\}$;
2. Ξ_2 defined as the black-box constraint set of the `camel six humps constrained` [156] benchmark optimization problem in Appendix B;
3. $\Xi_3 = \left\{\boldsymbol{x} : \|\boldsymbol{x} - \boldsymbol{x_c}\|_2^2 \geq r_{in}^2, \|\boldsymbol{x} - \boldsymbol{x_c}\|_2^2 \leq r_{out}^2\right\}$, where $\boldsymbol{x_c} = 0.5 \cdot \mathbf{1}_2, r_{in} = 0.25 \cdot r, r_{out} = 0.65 \cdot r$ and $r = 0.5 \cdot \|\mathbf{1}_2\|_2$;

4. $\Xi_4 = \left\{\boldsymbol{x} : \|\boldsymbol{x} - \boldsymbol{x}_{\boldsymbol{c}_1}\|_2^2 \leq r_1^2\right\} \cup \left\{\boldsymbol{x} : \|\boldsymbol{x} - \boldsymbol{x}_{\boldsymbol{c}_2}\|_2^2 \leq r_2^2\right\} \cup \{\boldsymbol{x} : 0.25 \cdot \mathbf{1}_2 \leq \boldsymbol{x} \leq 0.75 \cdot \mathbf{1}_2\}$, where $\boldsymbol{x}_{\boldsymbol{c}_1} = \begin{bmatrix}-0.5 & 0.5\end{bmatrix}^\top$, $r_1 = 0.5$, $\boldsymbol{x}_{\boldsymbol{c}_2} = \begin{bmatrix}0.75 & -0.75\end{bmatrix}^\top$ and $r_2 = 0.6$.

For each of the above Ξ-feasible regions, we generate a set $\mathcal{X}$ (2.9) of $N = 200$ samples, using a latin hypercube design (see Section 2.4), and evaluate the Ξ-feasibility of each point, obtaining the set $\mathcal{U}_\Xi$ in (2.11). Then, we estimate the probability of Ξ-feasibility as in (4.16). As it is common in machine learning, we define a threshold $\gamma = 0.5$ to classify between Ξ-feasible and Ξ-infeasible samples. In particular, if $p_N(\boldsymbol{x} \in \Xi) \geq \gamma$, we deem $\boldsymbol{x}$ as Ξ-feasible and vice-versa if $p_N(\boldsymbol{x} \in \Xi) < \gamma$. Figure 15 depicts the obtained results. Notice how the IDWI function is able to approximate even disconnected feasible regions.

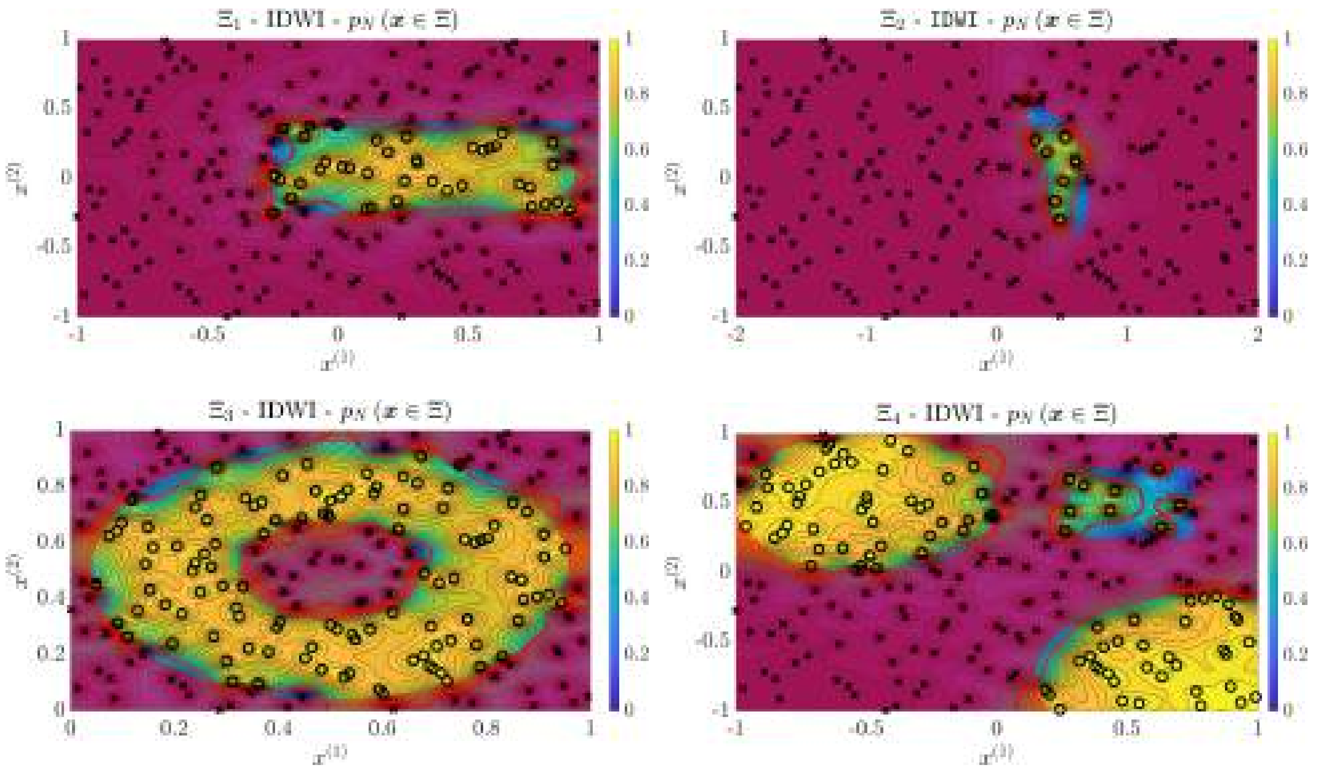


Figure 15: Probability of Ξ-feasibility estimated by the IDWI function in (4.16) **for different Ξ-feasible regions, described in Example 4.5. The Ξ-feasible samples are depicted as black circles whereas the Ξ-infeasible ones are black crosses. The shaded red areas denote the Ξ-infeasible regions. Finally, the red lines are the decision boundaries, i.e. $p_N(\boldsymbol{x} \in \Xi) = \gamma$, with $\gamma = 0.5$.**

4.2 Exploration functions

Exploration plays a key role in any global optimization procedure (see Chapters 1, 2 and 3). As a matter of fact, `GLIS` [10], `GLISp` [11], `C-GLIS` and `C-GLISp` [156] use acquisition functions that are weighted sums between the surrogate models in Section 4.1 and one or more exploration functions (see $a_N(\boldsymbol{x})$ in (2.58), (3.23) and (3.25)). In this Section, we review each exploration function in detail, starting from the IDW distance function.

Definition 4.3: Inverse distance weighting distance function [10]. *Consider the set of samples $\mathcal{X}$ in* (2.9)*. In* GLIS *[10] and* GLISp *[11], the* <u>*inverse distance weighting distance function,*</u> $z_N : \mathbb{R}^n \to (-1, 0]$*, is defined as follows*[a]*:*

$$z_N(\boldsymbol{x}) = \begin{cases} 0 & \text{if } \boldsymbol{x} \in \mathcal{X} \\ -\frac{2}{\pi} \cdot \arctan\left(\frac{1}{\sum_{i=1}^{N} w_i(\boldsymbol{x})}\right) & \text{otherwise} \end{cases}. \tag{4.17}$$

C-GLIS *and* C-GLISp *[156] use an alternative IDW distance function. In particular, consider the case $\boldsymbol{x} \in \mathbb{R}^n \setminus \mathcal{X}$, then $z_N : \mathbb{R}^n \to \mathbb{R}$ is defined as:*

$$\begin{aligned} z_N(\boldsymbol{x}) = & \left(\frac{N}{N_{max}} - 1\right) \cdot \arctan\left(\frac{\sum_{i=1, i \neq i_{best}(N)}^{N} w_i(\boldsymbol{x_{best}}(N))}{\sum_{i=1}^{N} w_i(\boldsymbol{x})}\right) + \\ & - \frac{N}{N_{max}} \cdot \arctan\left(\frac{1}{\sum_{i=1}^{N} w_i(\boldsymbol{x})}\right), \end{aligned} \tag{4.18}$$

while $z_N(\boldsymbol{x}) = 0, \forall \boldsymbol{x} \in \mathcal{X}$. Differently from $z_N(\boldsymbol{x})$ in (4.17)*, $z_N(\boldsymbol{x})$ in* (4.18) *also takes into account the best candidate $\boldsymbol{x_{best}}(N)$ (whose index is $i_{best}(N) \in \mathbb{N}, 1 \leq i_{best}(N) \leq N$) as well as the budget $N_{max} \geq N$.*

[a]To be precise, the multiplicative constant $\frac{2}{\pi}$ of $z_N(\boldsymbol{x})$ in (4.17) is not considered in GLISp [11], although it is present for GLIS [10].

Both $z_N(\boldsymbol{x})$ in (4.17) and $z_N(\boldsymbol{x})$ in (4.18) promote the exploration of the feasible region Ω of the GOP (2.1) based only on the locations of the samples in $\mathcal{X}$ (2.9). The rationale behind the IDW distance function in (4.18) is that, in the early iterations of C-GLIS and C-GLISp [156], it encourages the exploration of those regions of Ω further away from the current best candidate. Furthermore, at least empirically, $z_N(\boldsymbol{x})$ in (4.18) is better suited for escaping local minima of the GOP (2.1) compared to $z_N(\boldsymbol{x})$ in (4.17) [156]. Figure 16 compares the two different formulations in Definition 4.3.

In this book, we will thoroughly analyze the IDW distance function in (4.17). In particular, in Section 5.1, we will study its shortcomings whereas, in Section 5.2, we will propose a revisited infill sampling criterion that improves the exploratory capabilities of $z_N(\boldsymbol{x})$ in (4.17).

Next, we report the differentiability result shown in [10].

Proposition 4.4: Differentiability of the IDW distance function [10]. *The IDW distance function $z_N(\boldsymbol{x})$ in* (4.17) *is differentiable everywhere.*

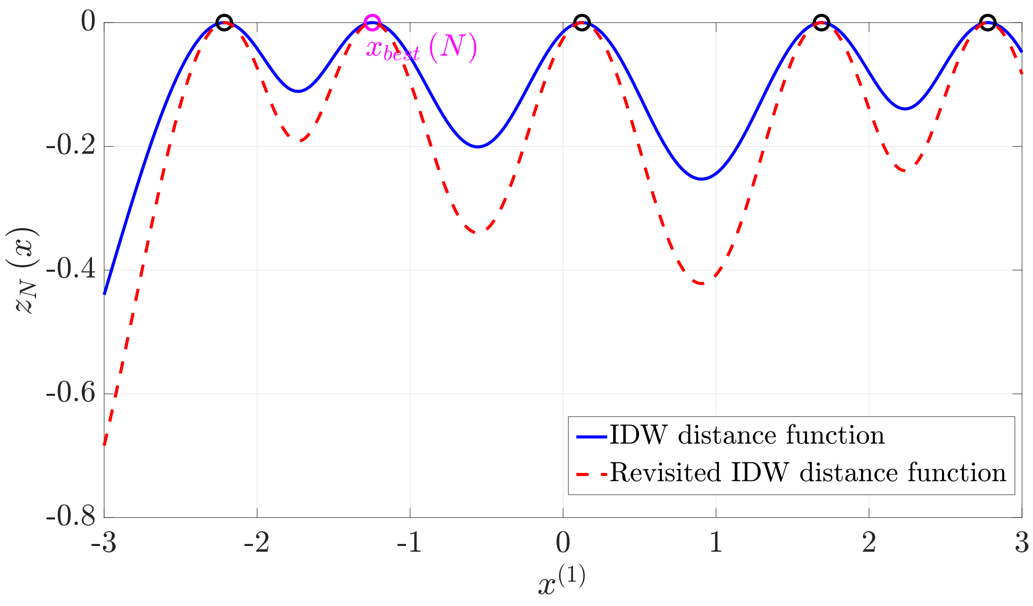


Figure 16: One-dimensional comparison between the two different formulations of the IDW distance function, namely $z_N(\boldsymbol{x})$ in (4.17) (blue line) and $z_N(\boldsymbol{x})$ in (4.18) (dashed red line). The samples in $\mathcal{X}, |\mathcal{X}| = N = 5$, are denoted using black circles, the only exception being the best candidate $x_{best}(N)$, which is reported in magenta. The budget for $z_N(\boldsymbol{x})$ in (4.18) is $N_{max} = 100$.

Proof. Let us consider the case $\boldsymbol{x} \in \mathbb{R}^n \setminus \mathcal{X}$. We have that:

$$\sum_{i=1}^{N} w_i(\boldsymbol{x}) \neq 0, \quad \forall \boldsymbol{x} \in \mathbb{R}^n \setminus \mathcal{X},$$

since all inverse distance weighting functions $w_i(\boldsymbol{x})$ in (4.11) are defined on that domain and only assume positive values. Furthermore, by Proposition 4.2, $w_i(\boldsymbol{x}), i = 1, \ldots, N$, are differentiable $\forall \boldsymbol{x} \in \mathbb{R}^n \setminus \mathcal{X}$ and so is their summation. Hence, due to the reciprocal rule, $z_N(\boldsymbol{x})$ in (4.17) is differentiable $\forall \boldsymbol{x} \in \mathbb{R}^n \setminus \mathcal{X}$, being the composition of functions that are differentiable on that domain. Now, consider the case $\boldsymbol{x} \in \mathcal{X}$. Let us compute the partial derivatives of $z_N(\boldsymbol{x})$ in (4.17) at any $\boldsymbol{x}_i \in \mathcal{X}$:

$$\begin{aligned}
\frac{\partial}{\partial x^{(j)}} z_N(\boldsymbol{x}_i) &= \lim_{t \to 0} \frac{z_N(\boldsymbol{x}_i + t \cdot \boldsymbol{e}_j) - z_N(\boldsymbol{x}_i)}{t} \\
&= \lim_{t \to 0} \frac{z_N(\boldsymbol{x}_i + t \cdot \boldsymbol{e}_j) - 0}{t} \\
&= -\frac{2}{\pi} \cdot \lim_{t \to 0} \frac{1}{t} \cdot \arctan\left(\frac{1}{\sum_{k=1}^{N} w_k(\boldsymbol{x}_i + t \cdot \boldsymbol{e}_j)} \right) \\
&= -\frac{2}{\pi} \cdot \lim_{t \to 0} \frac{1}{t} \cdot \arctan\left(\frac{1}{w_i(\boldsymbol{x}_i + t \cdot \boldsymbol{e}_j) + \sum_{k=1, k \neq i}^{N} w_k(\boldsymbol{x}_i + t \cdot \boldsymbol{e}_j)} \right) \\
&= -\frac{2}{\pi} \cdot \lim_{t \to 0} \frac{1}{t} \cdot \arctan\left(\frac{1}{\frac{1}{\|\cancel{\boldsymbol{x}_i} + t \cdot \boldsymbol{e}_j - \cancel{\boldsymbol{x}_i}\|_2^2} + \sum_{k=1, k \neq i}^{N} w_k(\boldsymbol{x}_i + t \cdot \boldsymbol{e}_j)} \right) \\
&= -\frac{2}{\pi} \cdot \lim_{t \to 0} \frac{1}{t} \cdot \arctan\left(\frac{1}{\frac{1}{t^2} + \sum_{k=1, k \neq i}^{N} w_k(\boldsymbol{x}_i + t \cdot \boldsymbol{e}_j)} \right)
\end{aligned}$$

$$
\begin{aligned}
&= -\frac{2}{\pi} \cdot \lim_{t \to 0} \frac{1}{t} \cdot \arctan \left(\frac{t^2}{1 + t^2 \cdot \sum_{k=1, k \neq i}^{N} w_k \left(\boldsymbol{x}_i + t \cdot \boldsymbol{e}_j \right)} \right) \\
&= -\frac{2}{\pi} \cdot \lim_{t \to 0} \frac{1}{t} \cdot \arctan \left(\frac{t^2}{1 + t^2 \cdot \sum_{k=1, k \neq i}^{N} \underbrace{\frac{1}{\|\boldsymbol{x}_i + t \cdot \boldsymbol{e}_j - \boldsymbol{x}_k\|_2^2}}_{>0 \text{ for } t \to 0, \boldsymbol{x}_i \neq \boldsymbol{x}_k}} \right) \\
&= 0.
\end{aligned}
$$

Next, define the first-order approximation of $z_N(\boldsymbol{x})$ in (4.17) near $\boldsymbol{x}_i \in \mathcal{X}$ (see Appendix A.3 and in particular (A.13)) as:

$$
L'(\boldsymbol{x}) = z_N(\boldsymbol{x}_i) + \boldsymbol{d}_{z_N}(\boldsymbol{x}_i)^\top \cdot (\boldsymbol{x} - \boldsymbol{x}_i),
$$

with:

$$
\boldsymbol{d}_{z_N}(\boldsymbol{x}_i) = \begin{bmatrix} \frac{\partial}{\partial x^{(1)}} z_N(\boldsymbol{x}_i) & \dots & \frac{\partial}{\partial x^{(n)}} z_N(\boldsymbol{x}_i) \end{bmatrix}^\top = \boldsymbol{0}_n,
$$

resulting in $L'(\boldsymbol{x}) = z_N(\boldsymbol{x}_i) = 0$. To prove the differentiability of $z_N(\boldsymbol{x})$ in (4.17) at $\boldsymbol{x}_i \in \mathcal{X}$, we need check whether:

$$
\lim_{\boldsymbol{x} \to \boldsymbol{x}_i} \frac{|z_N(\boldsymbol{x}) - L'(\boldsymbol{x})|}{\|\boldsymbol{x} - \boldsymbol{x}_i\|_2} = 0
$$

holds (see Definition A.25). By substituting to each function its expression we get:

$$
\begin{aligned}
\lim_{\boldsymbol{x} \to \boldsymbol{x}_i} \frac{|z_N(\boldsymbol{x}) - L'(\boldsymbol{x})|}{\|\boldsymbol{x} - \boldsymbol{x}_i\|_2} &= \lim_{\boldsymbol{x} \to \boldsymbol{x}_i} \frac{|z_N(\boldsymbol{x})|}{\|\boldsymbol{x} - \boldsymbol{x}_i\|_2} \\
&= \lim_{\boldsymbol{x} \to \boldsymbol{x}_i} \frac{\left| -\frac{2}{\pi} \cdot \arctan \left(\frac{1}{\sum_{k=1}^{N} w_k(\boldsymbol{x})} \right) \right|}{\|\boldsymbol{x} - \boldsymbol{x}_i\|_2} \\
&= \left| -\frac{2}{\pi} \right| \cdot \lim_{\boldsymbol{x} \to \boldsymbol{x}_i} \frac{\left| \overbrace{\arctan \left(\frac{1}{\sum_{k=1}^{N} w_k(\boldsymbol{x})} \right)}^{\geq 0, \forall \boldsymbol{x} \in \mathbb{R}^n} \right|}{\|\boldsymbol{x} - \boldsymbol{x}_i\|_2} \\
&= \frac{2}{\pi} \cdot \lim_{\boldsymbol{x} \to \boldsymbol{x}_i} \frac{\arctan \left(\frac{1}{\sum_{k=1}^{N} w_k(\boldsymbol{x})} \right)}{\|\boldsymbol{x} - \boldsymbol{x}_i\|_2}.
\end{aligned}
$$

Consider the case $\mathcal{X} = \{\boldsymbol{x}_1\}$, then it is easy to see that:

$$
\begin{aligned}
\lim_{\boldsymbol{x} \to \boldsymbol{x}_1} \frac{\arctan \left(\frac{1}{w_1(\boldsymbol{x})} \right)}{\|\boldsymbol{x} - \boldsymbol{x}_1\|_2} &= \lim_{\boldsymbol{x} \to \boldsymbol{x}_1} \frac{\arctan \left(\|\boldsymbol{x} - \boldsymbol{x}_1\|_2^2 \right)}{\|\boldsymbol{x} - \boldsymbol{x}_1\|_2} \\
&= \lim_{t \to 0} \frac{\arctan \left(t^2 \right)}{t} \\
&= 0.
\end{aligned}
$$

Similarly, for any $\mathcal{X} = \{\boldsymbol{x}_1, \dots, \boldsymbol{x}_N\}$, we have:

$$
\begin{aligned}
\lim_{\boldsymbol{x} \to \boldsymbol{x}_i} \frac{\arctan\left(\frac{1}{\sum_{k=1}^{N} w_k(\boldsymbol{x})}\right)}{\|\boldsymbol{x} - \boldsymbol{x}_i\|_2} &= \lim_{\boldsymbol{x} \to \boldsymbol{x}_i} \frac{\arctan\left(\frac{1}{\sum_{k=1}^{N} \frac{1}{\|\boldsymbol{x}-\boldsymbol{x}_k\|_2^2}}\right)}{\|\boldsymbol{x} - \boldsymbol{x}_i\|_2} \\
&= \lim_{\boldsymbol{x} \to \boldsymbol{x}_i} \frac{\arctan\left(\frac{1}{\frac{1}{\|\boldsymbol{x}-\boldsymbol{x}_i\|_2^2} + \sum_{k=1, k\neq i}^{N} \frac{1}{\|\boldsymbol{x}-\boldsymbol{x}_k\|_2^2}}\right)}{\|\boldsymbol{x} - \boldsymbol{x}_i\|_2} \\
&= \lim_{\boldsymbol{x} \to \boldsymbol{x}_i} \frac{\arctan\left[\frac{1}{\frac{1}{\|\boldsymbol{x}-\boldsymbol{x}_i\|_2^2} \cdot \left(1 + \sum_{k=1, k\neq i}^{N} \frac{\|\boldsymbol{x}-\boldsymbol{x}_i\|_2^2}{\|\boldsymbol{x}-\boldsymbol{x}_k\|_2^2}\right)}\right]}{\|\boldsymbol{x} - \boldsymbol{x}_i\|_2} \\
&= \lim_{\boldsymbol{x} \to \boldsymbol{x}_i} \frac{\arctan\left(\frac{\|\boldsymbol{x}-\boldsymbol{x}_i\|_2^2}{\underbrace{1 + \sum_{k=1, k\neq i}^{N} \frac{\|\boldsymbol{x}-\boldsymbol{x}_i\|_2^2}{\|\boldsymbol{x}-\boldsymbol{x}_k\|_2^2}}_{\to 1 \text{ for } \boldsymbol{x} \to \boldsymbol{x}_i, \boldsymbol{x}_i \neq \boldsymbol{x}_k}}\right)}{\|\boldsymbol{x} - \boldsymbol{x}_i\|_2} \\
&= 0.
\end{aligned}
$$

We have proven that the condition in Definition A.25 is satisfied. Therefore, $z_N(\boldsymbol{x})$ in (4.17) is differentiable at each $\boldsymbol{x}_i \in \mathcal{X}$.

In conclusion, by combining the results achieved for $\boldsymbol{x} \in \mathbb{R}^n \setminus \mathcal{X}$ and $\boldsymbol{x} \in \mathcal{X}$, we can state that $z_N(\boldsymbol{x})$ in (4.17) is differentiable everywhere. □

Algorithms `GLIS` [10] and `C-GLIS`, which are used for black-box optimization, also consider an additional exploration function that relies on the measures of the cost function in $\mathcal{Y}$ (2.10).

Definition 4.4: Inverse distance weighting variance function [10]. *Consider the set of samples $\mathcal{X}$ in* (2.9)*, the set of cost function measures $\mathcal{Y}$ in* (2.10) *and the surrogate model $\hat{f}_N(\boldsymbol{x})$ in* (4.1) *obtained as described in Section 4.1.1. The inverse distance weighting variance function, $s_N : \mathbb{R}^n \to \mathbb{R}$, is defined as follows:*

$$
s_N(\boldsymbol{x}) = -\sqrt{\sum_{i=1}^{N} v_i(\boldsymbol{x}) \cdot \left(y_i - \hat{f}_N(\boldsymbol{x})\right)^2}, \tag{4.19}
$$

where $v_i(\boldsymbol{x})$ is defined as in (4.14).

$s_N(\boldsymbol{x})$ in (4.19) represents the *confidence intervals* (with a negative sign) for the IDW interpolation function in (4.15) (i.e. if $\hat{f}_N(\boldsymbol{x})$ in (4.1) were to be replaced by (4.15)), see [69]. In GLIS [10] and C-GLIS, the IDW variance function is used to estimate the uncertainty of the RBF surrogate model in (4.1) at a point $\boldsymbol{x} \in \mathbb{R}^n$. Figure 17 depicts the confidence intervals for an interpolating and a non-interpolating surrogate (see Section 4.1.1). In the former case $s_N(\boldsymbol{x}_i) = 0, \forall \boldsymbol{x}_i \in \mathcal{X}$, whereas in the latter case we can have $s_N(\boldsymbol{x}_i) \neq 0$ for some $\boldsymbol{x}_i \in \mathcal{X}$.

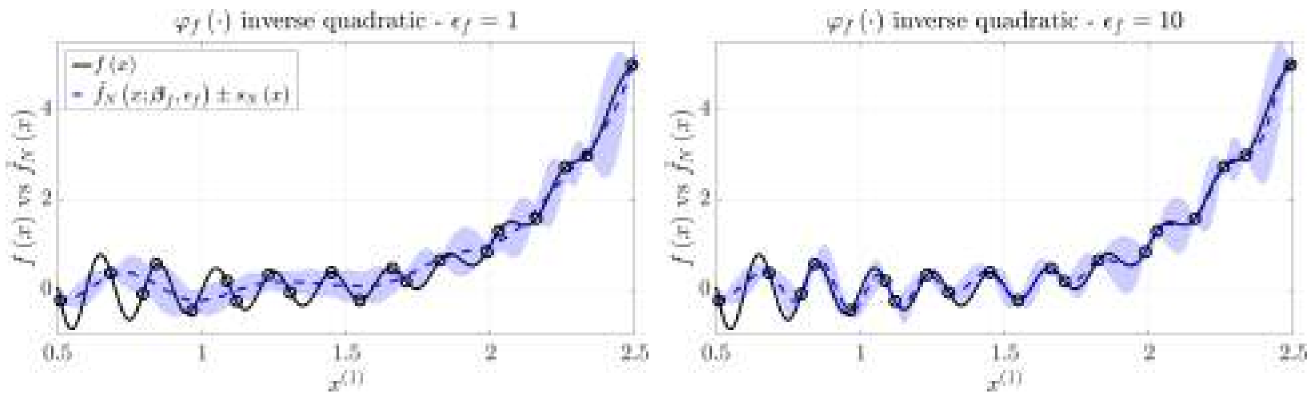


Figure 17: Confidence intervals for the surrogate model $\hat{f}_N\left(x;\boldsymbol{\beta}_f,\epsilon_f\right)$ in (4.1) (blue area), namely $\hat{f}_N\left(x;\boldsymbol{\beta}_f,\epsilon_f\right) \pm s_N(\boldsymbol{x})$, defined using an inverse quadratic radial basis function with $\epsilon_f = 1$ (left) and $\epsilon_f = 10$ (right). The real function $f(\boldsymbol{x})$ (black) is the one-dimensional gramacy and lee [53] function. The samples are generated as in Example 4.1.

The differentiability of the IDW variance function in (4.19) is addressed by the following Proposition.

Proposition 4.5: Differentiability of the IDW variance function [10]. *The IDW variance function $s_N(\boldsymbol{x})$ in (4.19) is differentiable everywhere.*

We omit the proof of the previous Proposition since it is not needed for the surrogate-based methods that we will propose in Chapter 5; the interested reader is referred to [10].

4.3 Infill sampling criteria

In this Section, we review the infill sampling criteria used by GLIS [10], GLISp [11], C-GLIS and C-GLISp [156]. All procedures look for new candidate samples by *minimizing acquisition functions that are weighted sums between the surrogate models in Section 4.1 and the exploration functions in Section 4.2.*

Definition 4.5: Acquisition functions for GLIS [10], GLISp [11], C-GLIS and C-GLISp [156]. *The aforementioned methods adopt infill sampling criteria that are based on the following* *acquisition functions $a_N : \mathbb{R}^n \to \mathbb{R}$:*

$$\begin{aligned}
&\texttt{GLIS [10]} && a_N(\boldsymbol{x}) = \hat{f}_N(\boldsymbol{x}) + \Delta Y \cdot \delta_1 \cdot z_N(\boldsymbol{x}) + \delta_2 \cdot s_N(\boldsymbol{x}), && \text{(4.20a)}\\
&\texttt{GLISp [11]} && a_N(\boldsymbol{x}) = \frac{\hat{f}_N(\boldsymbol{x})}{\Delta \hat{F}} + \delta \cdot z_N(\boldsymbol{x}), && \text{(4.20b)}\\
&\texttt{C-GLIS} && a_N(\boldsymbol{x}) = \hat{f}_N(\boldsymbol{x}) + \Delta Y \cdot \delta_1 \cdot z_N(\boldsymbol{x}) + \delta_2 \cdot s_N(\boldsymbol{x}) + && \text{(4.20c)}\\
& && \quad + \delta_\Xi \cdot [1 - p_N(\boldsymbol{x} \in \Xi)], && \\
&\texttt{C-GLISp [156]} && a_N(\boldsymbol{x}) = \frac{\hat{f}_N(\boldsymbol{x})}{\Delta \hat{F}} + \delta \cdot z_N(\boldsymbol{x}) + \delta_\Xi \cdot [1 - p_N(\boldsymbol{x} \in \Xi)]. && \text{(4.20d)}
\end{aligned}$$

$a_N(\boldsymbol{x})$ in (4.20a) *and in* (4.20b) *use the IDW distance function in* (4.17) *while the acquisition functions in* (4.20c) *and* (4.20b) *adopt $z_N(\boldsymbol{x})$ in* (4.18). *$\delta_1, \delta_2, \delta, \delta_\Xi \in \mathbb{R}_{\geq 0}$ are non-negative weights that define the exploration-exploitation trade-off as well as the penalization of those regions of Ω which are likely to contain Ξ-infeasible samples. Finally, $\Delta Y = \max\left\{\max_{y_i \in \mathcal{Y}} y_i - \min_{y_i \in \mathcal{Y}} y_i, \epsilon_{\Delta Y}\right\}$ (where $\epsilon_{\Delta Y} \in \mathbb{R}_{>0}$ is a small tolerance) and $\Delta \hat{F} = \max_{\boldsymbol{x}_i \in \mathcal{X}} \hat{f}_N(\boldsymbol{x}_i) - \min_{\boldsymbol{x}_i \in \mathcal{X}} \hat{f}_N(\boldsymbol{x}_i)$ are two scaling constants which ease the definition of the weights.*

We point out that *the exploration-explotation trade-off weights $\delta_1, \delta_2, \delta$ are kept fixed throughout the whole optimization processes of* `GLIS` *[10],* `GLISp` *[11],* `C-GLIS` *and* `C-GLISp` *[156].* In Section 5.2, we will propose a revisited infill sampling criterion that alternates between different values of the exploration-exploitation trade-off weights, giving rise to globally convergent unconstrained optimization procedures. Parameter δ_Ξ in (4.20c) and (4.20d) is recalibrated based on the standard deviation of the misclassification error of $p_N(\boldsymbol{x} \in \Xi) = p\left(\boldsymbol{x} \in \Xi \,\middle|\, \mathcal{U}_\Xi, \mathcal{X}, \boldsymbol{x}\right)$, denoted as $\hat{\delta}_\Xi \in \mathbb{R}_{\geq 0}$ and estimated through Leave-One-Out Cross-Validation (LOOCV) [58] as follows:

$$\hat{\delta}_\Xi = \min\left\{1, \sqrt{\frac{\sum_{i=1}^{N}\left[p\left(\boldsymbol{x}_i \in \Xi \,\middle|\, \mathcal{U}_\Xi \setminus \{u_i\}, \mathcal{X} \setminus \{\boldsymbol{x}_i\}, \boldsymbol{x}_i\right) - u_i\right]^2}{N-1}}\right\}. \tag{4.21}$$

Then, at each iteration of `C-GLIS` and `C-GLISp` [156], δ_Ξ in (4.20c) and (4.20d) is updated as:

$$\delta_\Xi = \left(1 - \hat{\delta}_\Xi\right) \cdot \delta_{\Xi,default}, \tag{4.22}$$

where $\delta_{\Xi,default} \in \mathbb{R}_{\geq 0}$ is a default value for the weight δ_Ξ chosen by the user.

The following Proposition covers the differentiability of the acquisition functions in (4.20a), (4.20b), (4.20c) and (4.20d).

Proposition 4.6: Differentiability of the acquisition functions in* (4.20a)*,* (4.20b)*,* (4.20c) *and* (4.20d)*. *The acquisition functions $a_N(\boldsymbol{x})$ in* (4.20a)*,* (4.20b)*,* (4.20c) *and* (4.20d) *are differentiable everywhere if and only if the chosen radial basis function $\phi_{f_i}(\boldsymbol{x};\epsilon_f) = \varphi_f(\epsilon_f \cdot \|\boldsymbol{x} - \boldsymbol{x}_i\|_2)$ for the surrogate model $\hat{f}_N\left(\boldsymbol{x};\boldsymbol{\beta}_f,\epsilon_f\right)$ in* (4.1) *is differentiable everywhere.*

Proof. The aforementioned result follows immediately from the application of Propositions 2.1, 4.3, 4.4 and 4.5, which cover the differentiability of each term that appears in the weighted sums in (4.20a), (4.20b), (4.20c) and (4.20d). □

For the sake of completeness, we stress that, in any case, the new candidate samples $\boldsymbol{x}_{N+1} \in \Omega$ are obtained by solving the following global optimization problem:

$$\begin{aligned} \boldsymbol{x}_{N+1} = \arg\min_{\boldsymbol{x}} a_N(\boldsymbol{x}) \\ \text{s.t.} \quad \boldsymbol{x} \in \Omega. \end{aligned} \tag{4.23}$$

4.4 Miscellaneous

There are other minor algorithmic details that characterize the GLIS [10], GLISp [11], C-GLIS and C-GLISp [156] procedures. In particular, the authors suggest to rescale all the decision variables of the GOP (2.1) so that they assume the same range. Furthermore, algorithms GLISp [11] and C-GLISp [156] also include a recalibration phase devoted to tuning the shape parameter ϵ_f of the RBF surrogate model in (4.1) (cf. Algorithm 5).

4.4.1 Rescaling of the decision vector

In [10], the author proposes to rescale the decision vector $\boldsymbol{x} \in \mathbb{R}^n$ of the GOP (2.1) so that all its entries $x^{(j)}$, $j = 1, \ldots, n$, assume the range $[-1, 1]$ (at least inside the feasible region Ω). Hereafter, we denote the rescaled decision vector as $\bar{\boldsymbol{x}} \in \mathbb{R}^n$. The rescaling procedure can be summarized as follows:

1. Obtain the upper and lower bounds $\boldsymbol{u_c}, \boldsymbol{l_c} \in \mathbb{R}^n$ for the bounding box of the constraint set Ω of the GOP (2.1) by solving the following $2 \cdot n$ optimization problems:

$$\begin{aligned} l_c^{(j)} = \min_{\boldsymbol{x}} \boldsymbol{e}_j^\top \cdot \boldsymbol{x} \\ \text{s.t.} \quad \boldsymbol{x} \in \Omega \quad \text{and} \end{aligned} \tag{4.24a}$$

$$\begin{aligned} u_c^{(j)} = &\max_{\boldsymbol{x}} \boldsymbol{e}_j^\top \cdot \boldsymbol{x} \\ \text{s.t.} \quad &\boldsymbol{x} \in \Omega, \\ \text{for} \quad &j = 1, \dots, n. \end{aligned} \tag{4.24b}$$

If Ω contains only linear equality and inequality constraints, then Problem (4.24a) and Problem (4.24b) are simple LPs.

2. Rescale each component $x^{(j)}, j = 1, \dots, n$, of $\boldsymbol{x}$ as:

$$\bar{x}^{(j)} = r_{\boldsymbol{x}\to\bar{\boldsymbol{x}}}^{(j)}\left(x^{(j)}\right) = \frac{2 \cdot x^{(j)} - \left(u_c^{(j)} + l_c^{(j)}\right)}{u_c^{(j)} - l_c^{(j)}}, \tag{4.25}$$

where $r_{\boldsymbol{x}\to\bar{\boldsymbol{x}}}^{(j)} : \mathbb{R} \to \mathbb{R}$ is the j-th *rescaling function*. To make the notation more compact, we define the vector-valued function $\boldsymbol{r}_{\boldsymbol{x}\to\bar{\boldsymbol{x}}} : \mathbb{R}^n \to \mathbb{R}^n$,

$$\boldsymbol{r}_{\boldsymbol{x}\to\bar{\boldsymbol{x}}}(\boldsymbol{x}) = \left[r_{\boldsymbol{x}\to\bar{\boldsymbol{x}}}^{(1)}\left(x^{(1)}\right) \quad \dots \quad r_{\boldsymbol{x}\to\bar{\boldsymbol{x}}}^{(n)}\left(x^{(n)}\right)\right]^\top,$$

and obtain the rescaled decision vector as:

$$\bar{\boldsymbol{x}} = \boldsymbol{r}_{\boldsymbol{x}\to\bar{\boldsymbol{x}}}(\boldsymbol{x}).$$

It is easy to prove that:

$$-\mathbf{1}_n \leq \boldsymbol{r}_{\boldsymbol{x}\to\bar{\boldsymbol{x}}}(\boldsymbol{x}) \leq \mathbf{1}_n, \quad \forall \boldsymbol{x} \in \Omega.$$

3. Rescale the constraint set Ω in (1.4) as:

$$\begin{aligned} \bar{\Omega} = \Big\{\bar{\boldsymbol{x}} : \bar{\boldsymbol{l}} \leq \bar{\boldsymbol{x}} \leq \bar{\boldsymbol{u}}, \bar{A}_{ineq} \cdot \bar{\boldsymbol{x}} \leq \bar{\boldsymbol{b}}_{\boldsymbol{ineq}}, \bar{A}_{eq} \cdot \bar{\boldsymbol{x}} = \bar{\boldsymbol{b}}_{\boldsymbol{eq}}, \\ \bar{\boldsymbol{g}}_{\boldsymbol{ineq}}(\bar{\boldsymbol{x}}) \leq \mathbf{0}_{q_3}, \bar{\boldsymbol{g}}_{\boldsymbol{eq}}(\bar{\boldsymbol{x}}) = \mathbf{0}_{q_4}\Big\}, \end{aligned}$$

where:

- The bounds are simply $\bar{\boldsymbol{l}} = -\mathbf{1}_n$ and $\bar{\boldsymbol{u}} = \mathbf{1}_n$;
- Matrices and vectors related to the linear constraints (both equality and inequality), say A and $\boldsymbol{b}$, are rescaled as:

$$\begin{aligned} \bar{A} &= A \cdot \operatorname{diag}\left\{\frac{u_c^{(1)} - l_c^{(1)}}{2}, \dots, \frac{u_c^{(n)} - l_c^{(n)}}{2}\right\}, \\ \bar{\boldsymbol{b}} &= \boldsymbol{b} - \frac{1}{2} \cdot A \cdot (\boldsymbol{u_c} + \boldsymbol{l_c}); \end{aligned}$$

- The nonlinear constraints functions become $\bar{\boldsymbol{g}}(\bar{\boldsymbol{x}}) = \boldsymbol{g}(\boldsymbol{r}_{\bar{\boldsymbol{x}}\to\boldsymbol{x}}(\bar{\boldsymbol{x}}))$, where $\boldsymbol{r}_{\bar{\boldsymbol{x}}\to\boldsymbol{x}} : \mathbb{R}^n \longrightarrow \mathbb{R}^n$ is a vector-valued function that scales $\bar{\boldsymbol{x}}$ back to the original range, as opposed to (4.25). In particular, $\boldsymbol{r}_{\bar{\boldsymbol{x}}\to\boldsymbol{x}}(\bar{\boldsymbol{x}})$ is defined as:

$$\boldsymbol{r}_{\bar{\boldsymbol{x}}\to\boldsymbol{x}}(\bar{\boldsymbol{x}}) = \left[r^{(1)}_{\bar{\boldsymbol{x}}\to\boldsymbol{x}}\left(\bar{x}^{(1)}\right) \quad \dots \quad r^{(n)}_{\bar{\boldsymbol{x}}\to\boldsymbol{x}}\left(\bar{x}^{(n)}\right)\right]^\top,$$

where:

$$r^{(j)}_{\bar{\boldsymbol{x}}\to\boldsymbol{x}}\left(\bar{x}^{(j)}\right) = \frac{\bar{x}^{(j)} \cdot \left(u_c^{(j)} - l_c^{(j)}\right) + \left(u_c^{(j)} + l_c^{(j)}\right)}{2}, \quad j = 1, \dots, n. \tag{4.26}$$

The rescaling of the optimization problem is carried out only once, at the start of the `GLIS` [10], `GLISp` [11], `C-GLIS` and `C-GLISp` [156] procedures. The rescaled GOP (2.1) is:

$$\begin{aligned} \bar{\mathcal{X}}^* = \arg &\min_{\bar{\boldsymbol{x}}} \bar{f}(\bar{\boldsymbol{x}}) \\ \text{s.t.} \quad &\bar{\boldsymbol{x}} \in \bar{\Omega} \cap \bar{\Xi}, \end{aligned} \tag{4.27}$$

where:

- The black-box cost function (or the scoring function of the human decision-maker) is $\bar{f}(\bar{\boldsymbol{x}}) = f(\boldsymbol{r}_{\bar{\boldsymbol{x}}\to\boldsymbol{x}}(\bar{\boldsymbol{x}}))$. Furthermore, in the preference-based optimization framework, the preference function in (3.8) becomes:

$$\bar{\pi}_{\gtrsim}(\bar{\boldsymbol{x}}_i, \bar{\boldsymbol{x}}_j) = \begin{cases} -1 & \text{if } \bar{f}(\bar{\boldsymbol{x}}_i) < \bar{f}(\bar{\boldsymbol{x}}_j) \\ 0 & \text{if } \bar{f}(\bar{\boldsymbol{x}}_i) = \bar{f}(\bar{\boldsymbol{x}}_j) \\ 1 & \text{if } \bar{f}(\bar{\boldsymbol{x}}_i) > \bar{f}(\bar{\boldsymbol{x}}_j) \end{cases}.$$

- The Ξ-feasibility function in (2.4) becomes $\bar{u}_{\bar{\Xi}}(\bar{\boldsymbol{x}}) = u_{\Xi}(\boldsymbol{r}_{\bar{\boldsymbol{x}}\to\boldsymbol{x}}(\bar{\boldsymbol{x}}))$ and, consequently, we can define the Ξ-feasible region as $\bar{\Xi} = \left\{\bar{\boldsymbol{x}} : \bar{u}_{\bar{\Xi}}(\bar{\boldsymbol{x}}) = 1\right\}$. Similarly, if the black-box constraints functions in (2.2) are measurable, we define $\bar{\boldsymbol{g}}_{\bar{\Xi}}(\bar{\boldsymbol{x}}) = \boldsymbol{g}_{\Xi}(\boldsymbol{r}_{\bar{\boldsymbol{x}}\to\boldsymbol{x}}(\bar{\boldsymbol{x}}))$ and specify $\bar{\Xi}$ accordingly.

- The global minimizers of the GOP (2.1), i.e. $\boldsymbol{x}_i^* \in \mathcal{X}^*$ in (2.3), are rescaled, obtaining:

$$\bar{\mathcal{X}}^* = \left\{\bar{\boldsymbol{x}}_i^* : \bar{\boldsymbol{x}}_i^* = \boldsymbol{r}_{\boldsymbol{x}\to\bar{\boldsymbol{x}}}\left(\boldsymbol{x}_i^*\right), \boldsymbol{x}_i^* \in \mathcal{X}^*, i = 1, \dots, N^*\right\}.$$

Clearly, if the rescaling is performed, then the surrogates in Section 4.1 and the exploration functions in Section 4.2 are built considering the rescaled set of samples,

$$\bar{\mathcal{X}} = \{\bar{\boldsymbol{x}}_i : i = 1, \dots, N, \bar{\boldsymbol{x}}_i = \boldsymbol{r}_{\boldsymbol{x}\to\bar{\boldsymbol{x}}}(\boldsymbol{x}_i), \boldsymbol{x}_i \in \mathcal{X}\},$$

instead of $\mathcal{X}$ in (2.9).

We conclude this Section by pointing out the relationship between the rescaling in (4.25) and *automatic relevance determination*, a strategy that is widely used in Gaussian process regression to weigh differently the variables $x^{(j)}, j = 1, \dots, n$, in order to improve the quality of the surrogate model in (2.46) [15, 153].

Remark 4.1 (Relationship to automatic relevance determination)**.** *Many of the radial basis functions in Definition 2.6 depend on the squared distance between two points. To analyze the effects of the rescaling in* (4.25)*, consider the squared Euclidean distance between a point $\bar{\boldsymbol{x}} \in \bar{\Omega}$ and a sample $\bar{\boldsymbol{x}}_i \in \bar{\mathcal{X}}$. We have:*

$$\begin{aligned}
\|\bar{\boldsymbol{x}} - \bar{\boldsymbol{x}}_i\|_2^2 &= (\bar{\boldsymbol{x}} - \bar{\boldsymbol{x}}_i)^T (\bar{\boldsymbol{x}} - \bar{\boldsymbol{x}}_i) \\
&= \sum_{j=1}^{n} \left(\bar{x}^{(j)} - \bar{x}_i^{(j)}\right)^2 \\
&= \sum_{j=1}^{n} \left(\frac{2 \cdot x^{(j)} - \left(u_c^{(j)} + l_c^{(j)}\right)}{u_c^{(j)} - l_c^{(j)}} - \frac{2 \cdot x_i^{(j)} - \left(u_c^{(j)} + l_c^{(j)}\right)}{u_c^{(j)} - l_c^{(j)}} \right)^2 \\
&= \sum_{j=1}^{n} \left[\frac{2}{u_c^{(j)} - l_c^{(j)}} \cdot \left(x^{(j)} - x_i^{(j)}\right) \right]^2 .
\end{aligned} \tag{4.28}$$

We can see that the distances $x^{(j)} - x_i^{(j)}$ are scaled by constant terms, namely $\theta_f^{(j)} = \frac{2}{u_c^{(j)} - l_c^{(j)}}, \theta_f^{(j)} \in \mathbb{R}_{>0}$. Similarly, in Gaussian process regression, a commonly used kernel is the squared exponential kernel, which can be defined as [153]:

$$k\left(\boldsymbol{x}, \boldsymbol{x}_i; \boldsymbol{\theta}_f\right) = \theta_f^{(0)} \cdot \exp\left\{ -\frac{1}{2} \cdot \sum_{j=1}^{n} \left[\theta_f^{(j)} \cdot \left(x^{(j)} - x_i^{(j)}\right)^2 \right] \right\}, \tag{4.29}$$

where $\boldsymbol{\theta}_f = \begin{bmatrix} \theta_f^{(0)} & \theta_f^{(1)} & \dots & \theta_f^{(n)} \end{bmatrix}^\top \in \mathbb{R}^{n+1}$. With a slight abuse of notation, $\theta_f^{(0)}$ is an additional hyper-parameter that is not related to a specific dimension of the variable $\boldsymbol{x}$. Typically, in the $\mathcal{GP}$ framework, $\boldsymbol{\theta}_f$ is computed by maximizing the marginal likelihood as in Problem (2.41)*. Clearly, if $\theta_f^{(j)}$ is small for any $j = 1, \dots, N$, then the surrogate in* (2.46) *with the kernel in* (4.29) *becomes relatively insensitive to the variable $x^{(j)}$. For the latter reason, this approach is called <u>automatic relevance determination</u> [15, 153]. Vice-versa, selecting $\theta_f^{(j)}, j = 1, \dots, N$, as in* (4.28) *is computationally lighter and makes each variable $x^{(j)}, j = 1, \dots, n$, equally as relevant.*

4.4.2 Recalibration of the surrogate models

The quality of the surrogate model $\hat{f}_N\left(\boldsymbol{x}; \boldsymbol{\beta}_f, \epsilon_f\right)$ in (4.1) depends on the choice of the radial function $\varphi_f(\cdot)$ as well as on the value of the shape parameter ϵ_f, which need to be properly selected. A common way to tune these hyper-parameters in the machine learning framework is to use cross-validation [58]. In particular, we employ K-fold grid search cross-validation to choose the best model among a set of alternatives. Consider the set of $N_{models} \in \mathbb{N}$ models:

$$\mathcal{M}_f = \left\{\varphi_{f_1}(\cdot), \ldots, \varphi_{f_{N_{models}}}(\cdot)\right\}, \tag{4.30}$$

where each model $\varphi_{f_m}(\cdot), m = 1, \ldots, N_{models}$, is associated to a certain radial function (such as the ones in Definition 2.6) and shape parameter ϵ_{f_m}. Usually, as it has been done in GLISp [11], the radial function is kept constant and the only varying parameter is ϵ_f. However, we cover this topic more generally, allowing $\varphi_{f_m}(\cdot)$ to vary as well. Furthermore, in [11], the authors only address the recalibration of the shape parameter for PBO. In this Section, we also give some insights on how to tune ϵ_f for BBO, see [24, 119].

K-fold grid search cross-validation operates as follows:

1. Build the dataset $\mathcal{D}$ which, in the black-box optimization framework, is equal to (see (2.9) and (2.10)):

   $$\mathcal{D} = \{(\boldsymbol{x}_i, y_i) : \boldsymbol{x}_i \in \mathcal{X}, y_i \in \mathcal{Y}, i = 1, \ldots, N\}, \tag{4.31}$$

   while, in the preference-based case, it amounts to (see (2.9), (3.9) and (3.10)):

   $$\mathcal{D} = \Big\{ \left(\boldsymbol{x}_{\ell(h)}, \boldsymbol{x}_{\kappa(h)}, b_h\right) : \boldsymbol{x}_{\ell(h)}, \boldsymbol{x}_{\kappa(h)} \in \mathcal{X}, b_h \in \mathcal{B}, (\ell(h), \kappa(h)) \in \mathcal{S}, \\ b_h = \pi_{\succsim}\left(\boldsymbol{x}_{\ell(h)}, \boldsymbol{x}_{\kappa(h)}\right), h = 1, \ldots, M\Big\}. \tag{4.32}$$

   We also define the number of elements in $\mathcal{D}$ as $N_{\mathcal{D}} = |\mathcal{D}|$, which are N for BBO and M for PBO.

2. Rather than explicitly defining the number of folds, $K \in \mathbb{N}$, which $\mathcal{D}$ needs to be split into, we choose a ratio $R_f \in [0, 1]$. Then, the maximum number of elements for each fold is selected as:

   $$N_{\mathcal{D}_{fold}} = \max\left\{\lfloor R_f \cdot N_{\mathcal{D}} \rfloor, 1\right\},$$

   whereas the number of folds is:

   $$K = \left\lceil \frac{N_{\mathcal{D}}}{N_{\mathcal{D}_{fold}}} \right\rceil.$$

   Notice that $R_f = 1$ equates to not performing any cross-validation while $R_f = 0$ results in leave-one-out cross-validation.

3. Randomly partition $\mathcal{D}$ into K (almost) equally-sized subsets (folds) $\mathcal{D}^{(1)}, \ldots, \mathcal{D}^{(K)}$, i.e. such that:

   $$\mathcal{D} = \bigcup_{k=1}^{K} \mathcal{D}^{(k)}, \quad \mathcal{D}^{(i)} \cap \mathcal{D}^{(j)} = \emptyset, \forall i, j = 1, ..., K, i \neq j,$$

and $\left|\mathcal{D}^{(k)}\right| \leq N_{\mathcal{D}_{fold}}, \forall k = 1, \dots, K$.

4. For each model $\varphi_{f_m}(\cdot) \in \mathcal{M}_f$, build K surrogates $\hat{f}_m^{(k)}(\cdot)$[1], $k = 1, \dots, K$, by computing $\boldsymbol{\beta}_f$ using either (4.4) (BBO) or (4.10) (PBO) and starting from the *training sets* $\mathcal{D}_{train}^{(k)} = \bigcup_{j \neq k} \mathcal{D}^{(j)}$.

5. Assess the quality of each surrogate on the *validation sets* $\mathcal{D}_{val}^{(k)} = \mathcal{D}^{(k)}$, i.e. the folds not used during the training phase ($\mathcal{D}_{train}^{(k)} \cap \mathcal{D}_{val}^{(k)} = \emptyset, \mathcal{D}_{train}^{(k)} \cup \mathcal{D}_{val}^{(k)} = \mathcal{D}, \forall k = 1, \dots, K$). To do so, a proper figure of merit or cost function, $J_{R_f}\left(\hat{f}_m^{(k)}(\cdot), \mathcal{D}_{val}^{(k)}\right)$, needs to be used:

   - For black-box optimization, a common choice is the *Root Mean Square Error (RMSE)* [24, 119], i.e.:

   $$J_{R_f}\left(\hat{f}_m^{(k)}(\cdot), \mathcal{D}_{val}^{(k)}\right) = \sqrt{\frac{1}{\left|\mathcal{D}_{val}^{(k)}\right|} \cdot \sum_{(\boldsymbol{x}_i, y_i) \in \mathcal{D}_{val}^{(k)}} \left[\hat{f}_m^{(k)}(\boldsymbol{x}_i) - y_i\right]^2}. \tag{4.33}$$

   - In the context of preference-based optimization, the authors of `GLISp` [11] propose to use the *number of correctly classified preferences*, which is defined as:

   $$J_{R_f}\left(\hat{f}_m^{(k)}(\cdot), \mathcal{D}_{val}^{(k)}\right) = \sum_{\left(\boldsymbol{x}_{\ell(h)}, \boldsymbol{x}_{\kappa(h)}, b_h\right) \in \mathcal{D}_{val}^{(k)}} \mathbb{I}\left(\hat{\pi}_{\succsim_m}^{(k)}\left(\boldsymbol{x}_{\ell(h)}, \boldsymbol{x}_{\kappa(h)}\right) = b_h\right), \tag{4.34}$$

   where $\hat{\pi}_{\succsim_m}^{(k)}(\cdot, \cdot)$ is the surrogate preference function defined from $\hat{f}_m^{(k)}(\cdot)$ (see (4.5)) and $\mathbb{I}(\cdot)$ is the indicator function (which assumes value one whenever the preference b_h is classified correctly by $\hat{\pi}_{\succsim_m}^{(k)}(\cdot, \cdot)$ and zero otherwise).

6. Compute the average performances of each model $\varphi_{f_m}(\cdot) \in \mathcal{M}_f$, i.e.:

$$J_{R_f}^{avg}\left(\varphi_{f_m}(\cdot)\right) = \frac{\sum_{k=1}^{K} J_{R_f}\left(\hat{f}_m^{(k)}(\cdot), \mathcal{D}_{val}^{(k)}\right)}{K}. \tag{4.35}$$

7. Select the best model among the ones in $\mathcal{M}_f$, i.e. $\varphi_{f_m}(\cdot)$ that either minimizes the average RMSE or maximizes the average number of correctly classified preferences.

Concerning the black-box optimization case, or rather the more general context of approximating an unknown function $f(\boldsymbol{x})$ using the radial basis function expansion in (4.1), there are alternative ways of selecting the shape parameter ϵ_f. For example, in [24, 119], ϵ_f is chosen by solving an optimization problem whose cost function is a norm of the vector whose entries are the RMSEs in (4.33) obtained via LOOCV.

[1]For ease of notation and clarity, we use m (i.e. the index of the model in $\mathcal{M}_f$) as the subscript of $\hat{f}_m^{(k)}(\cdot)$ instead of the number of samples used for its construction, which are $\left|\mathcal{D}_{train}^{(k)}\right|$.

In the preference-based framework, as pointed out in [11], we are mostly interested in correctly predicting those preferences that concern the current best candidate $\boldsymbol{x_{best}}(N)$. Therefore, all the preferences associated to such sample should always be present in the training sets and never in the validation sets. This is also necessary to properly build the weight vector $\boldsymbol{r_{QP}}$ of Problem (4.10) as in (4.7). Thus, we define the set:

$$\mathcal{D}_{best} = \left\{ \left(\boldsymbol{x}_{\ell(h)}, \boldsymbol{x}_{\kappa(h)}, b_h \right) \in \mathcal{D} : \boldsymbol{x}_{\ell(h)} = \boldsymbol{x_{best}}(N) \text{ or } \boldsymbol{x}_{\kappa(h)} = \boldsymbol{x_{best}}(N) \right\} \tag{4.36}$$

and obtain the K folds from $\mathcal{D} \setminus \mathcal{D}_{best}$ instead of $\mathcal{D}$ in (4.32). Finally, set $\mathcal{D}_{best}$ is always included inside the training sets $\mathcal{D}_{train}^{(k)}$, $k = 1, \ldots, K$.

The whole K-fold grid search cross-validation procedure for BBO and PBO is formalized in Algorithm 8.

Finally, we show an Example where we apply Algorithm 8 to select both the radial function and the shape parameter of $\hat{f}_N\left(\boldsymbol{x}; \boldsymbol{\beta}_f, \epsilon_f\right)$ in (4.1).

Example 4.6: Recalibration of the surrogate of $f(\boldsymbol{x})$

Consider the `camel six humps` [62] benchmark function defined as in Example 4.2. We use the same set of samples $\mathcal{X}$ (2.9) generated in Example 4.2 to recalibrate the hyper-parameters of the surrogate model $\hat{f}_N\left(\boldsymbol{x}; \boldsymbol{\beta}_f, \epsilon_f\right)$ in (4.1), both in the black-box and in the preference-based frameworks. We consider two different sets of models $\mathcal{M}_f$ in (4.30):

- The first one is built by keeping the radial function $\varphi_{f_m}(\cdot)$ fixed (inverse quadratic) and varying the shape parameter ϵ_f uniformly, in a logarithmically spaced grid, between $\epsilon_f = 10^{-2}$ and $\epsilon_f = 10^1$ (including also $\epsilon_f = 1$). The total number of models is $N_{models} = 21$;
- The second one is aimed at evaluating the performances of different RBFs, keeping the shape parameter constant and equal to $\epsilon_f = 1$. The selection is made between the $N_{models} = 8$ radial basis functions in Definition 2.6.

The remaining hyper-parameters are selected as:

- BBO: $\epsilon_{SVD} = 10^{-6}$ for the low-rank approximation in (4.4);
- PBO: $\lambda_f = 10^{-6}, \sigma_\pi = 10^{-2}$ for Problem (4.10);
- $R_f = 0$, i.e. we perform leave-one-out grid search cross-validation.

Figure 18 shows the average performances of each model in $\mathcal{M}_f$, i.e. $J_{R_f}^{avg}\left(\varphi_{f_m}(\cdot)\right)$ in (4.35), as well as the best models found by Algorithm 8. Notice how all the resulting surrogates capture where the most promising region of $f(\boldsymbol{x})$ is located, which is roughly $-0.5 \leq x^{(1)} \leq 0.5$. Furthermore, we can see that, *at least in the BBO framework, the shape parameter ϵ_f has a higher impact on the performances compared to the choice of the RBF.*

Algorithm 8: K-fold grid search cross-validation for RBF surrogates of $f(\boldsymbol{x})$

Input: (i) Dataset $\mathcal{D}$ defined as in (4.31) or (4.32); (ii) Set of models $\mathcal{M}_f$ in (4.30); (iii) Ratio $R_f \in [0, 1]$ which defines the dimension of the folds of $\mathcal{D}$.

Output: (i) Best model $\varphi_{f_{best}}(\cdot)$.

```
1: In the preference-based case, obtain the dataset $\mathcal{D}_{best}$ as in (4.36), while for the black-box case simply set $\mathcal{D}_{best} = \emptyset$
2: (Possibly) remove the preferences associated to $\boldsymbol{x_{best}}(N)$ in $\mathcal{D}$: $\mathcal{D} = \mathcal{D} \setminus \mathcal{D}_{best}$
3: Get the cardinality of $\mathcal{D}$: $N_{\mathcal{D}} = |\mathcal{D}|$
4: Compute the maximum number of elements for each fold: $N_{\mathcal{D}_{fold}} = \max\left\{\lfloor R_f \cdot N_{\mathcal{D}} \rfloor, 1\right\}$
5: Compute the number of folds: $K = \left\lceil \frac{N_{\mathcal{D}}}{N_{\mathcal{D}_{fold}}} \right\rceil$
6: Randomly divide the dataset $\mathcal{D}$ into $K$ (nearly equally-sized) disjoint subsets (folds), namely $\mathcal{D}^{(1)}, \dots, \mathcal{D}^{(K)}$
7: for $k = 1$ to $K$ do
8:     Build the training set as $\mathcal{D}_{train}^{(k)} = \left[\bigcup_{j \neq k} \mathcal{D}^{(j)}\right] \cup \mathcal{D}_{best}$
9:     Select the validation set as $\mathcal{D}_{val}^{(k)} = \mathcal{D}^{(k)}$
10:    for $m = 1$ to $N_{models}$ do
11:        Fit the surrogate $\hat{f}_m^{(k)}(\cdot)$ in (4.1), associated to the model $\varphi_{f_m}(\cdot) \in \mathcal{M}_f$, on the training data $\mathcal{D}_{train}^{(k)}$ by computing $\boldsymbol{\beta}_f$ either using (4.4) (BBO) or (4.10) (PBO)
12:        Evaluate the performances of $\hat{f}_m^{(k)}(\cdot)$ on the validation set $\mathcal{D}_{val}^{(k)}$, i.e. compute $J_{R_f}\left(\hat{f}_m^{(k)}(\cdot), \mathcal{D}_{val}^{(k)}\right)$ as in (4.33) (BBO) or in (4.34) (PBO)
13: Compute the average performances $J_{R_f}^{avg}\left(\varphi_{f_m}(\cdot)\right), \forall m = 1, \dots, N_{models}$, as in (4.35)
14: Select the best model, $\varphi_{f_{best}}(\cdot)$, as the one that minimizes (BBO) or maximizes (PBO) $J_{R_f}^{avg}\left(\varphi_{f_m}(\cdot)\right)$
```

4.5 Algorithms

Algorithms 10, 11 and 12 describe, respectively, the `GLIS` [10], `GLISp` [11] and `C-GLISp` [156] procedures in detail. We do not report the `C-GLIS` method since it can easily be obtained by "combining" Algorithm 10 with Algorithm 12, keeping in mind that the acquisition function is the one in (4.20c) instead of (4.20a)/(4.20d). All the procedures follow the surrogate-based scheme described by Algorithm 5, although they include an additional phase devoted to rescaling the GOP (2.1) (as proposed in Section 4.4.1). In Algorithm 9, we formalize what it means for a new candidate sample $\boldsymbol{x}_{N+1} \in \Omega$, obtained as described in Section 4.3, to improve upon the current best candidate $\boldsymbol{x_{best}}(N)$. We point out that no recalibration of the surrogate model $\hat{f}_N(\boldsymbol{x})$ in (4.1) is performed by the `GLIS` [10] and `C-GLIS` procedures. Instead, algorithms `GLISp` [11] and `C-GLISp` [156] recalibrate only the shape parameter ϵ_f of $\hat{f}_N(\boldsymbol{x})$ in (4.1) through K-fold grid search cross-validation, as described in Section 4.4.2. In practice, ϵ_f is tuned only at certain iterations of the GLISp [11] and C-GLISp [156] algorithms, as highlighted by the set $\mathcal{K}_{R_f} \subseteq \{1, \dots, N_{max} - N_{init}\}$.

4.6 Chapter summary

In this Chapter, we have thoroughly analyzed the GLIS [10], GLISp [11], C-GLIS and C-GLISp [156] procedures. All methods use the radial basis function expansion surrogate to approximate either the black-box cost function or the scoring function of the human decision-maker. The weights for the latter model are either estimated by imposing the interpolation conditions (BBO), solving the corresponding linear system using a low-rank approximation of the interpolation matrix, or by solving a convex quadratic program, which ensures that the surrogate model "captures" the preferences expressed by the decision-maker (PBO). For what concerns the black-box constraints functions, C-GLIS and C-GLISp [156] rely on the inverse distance weighting interpolation function to approximate the probability of Ξ-feasibility, using only the information contained inside the sets $\mathcal{X}$ in (2.9) and $\mathcal{U}_\Xi$ in (2.11). All methods look for new candidate samples by minimizing acquisition functions that are weighted sums between the surrogate models and, at most, two exploration functions. At the moment, no proofs of global convergence are available for any of the aforementioned algorithms. In Chapter 5, we will propose globally convergent extensions of GLIS [10] and GLISp [11]. Instead, in Chapter 6, we will extend algorithms C-GLIS and C-GLISp [156] to make them more sample efficient.

Algorithm 9: Check for improvement of $\boldsymbol{x_{best}}(N)$

Input: (i) Current best candidate $\boldsymbol{x_{best}}(N) \in \Omega$ (ii) New candidate sample $\boldsymbol{x}_{N+1} \in \Omega$; (iii) Information obtained by the sample evaluations of $\boldsymbol{x_{best}}(N)$ and $\boldsymbol{x}_{N+1}$: (BBO) measures of the black-box cost function at $\boldsymbol{x_{best}}(N)$ and at $\boldsymbol{x}_{N+1}$, namely $y_{best}(N)$ and y_{N+1}, (PBO) preference expressed by the DM, i.e. $b_{M+1} = \pi_{\gtrsim}(\boldsymbol{x}_{N+1}, \boldsymbol{x_{best}}(N))$, (BBO and PBO) Ξ-feasibility information $u_{best}(N) = u_\Xi(\boldsymbol{x_{best}}(N))$ and $u_{N+1} = u_\Xi(\boldsymbol{x}_{N+1})$. If no black-box constraints are present, then $u_{best}(N) = u_{N+1} = 1$.

Output: (i) Flag $hasImproved$, which indicates whether or not $\boldsymbol{x}_{N+1}$ improves upon $\boldsymbol{x_{best}}(N)$.

```
1: if $u_{best}(N) = u_{N+1}$ then
2:     if either $y_{N+1} < y_{best}(N)$ (in BBO) or $\boldsymbol{x}_{N+1} \succ \boldsymbol{x_{best}}(N)$ (i.e. $b_{M+1} = -1$, in PBO) then
3:         $hasImproved$ = true
4:     else
5:         $hasImproved$ = false
6: else
7:     if $\boldsymbol{x}_{N+1} \in \Xi$ and $\boldsymbol{x}_{best}(N) \notin \Xi$ (i.e. $u_{N+1} = 1, u_{best}(N) = 0$) then
8:         $hasImproved$ = true        ▹ A $\Xi$-feasible sample is always better than a $\Xi$-infeasible one
9:     else
10:        $hasImproved$ = false
```

Algorithm 10: `GLIS` [10]

Input: (i) A-priori known constraint set Ω of the GOP (2.1); (ii) Number of initial samples $N_{init} \in \mathbb{N}$; (iii) Budget $N_{max} \in \mathbb{N}, N_{max} > N_{init}$; (iv) Hyper-parameters for the surrogate model $\hat{f}_N(\boldsymbol{x})$ in (4.1), namely shape parameter $\epsilon_f \in \mathbb{R}_{>0}$, radial function $\varphi_f(\cdot)$ and threshold $\epsilon_{SVD} \in \mathbb{R}_{>0}$; (v) Exploration-exploitation trade-off weights $\delta_1, \delta_2 \in \mathbb{R}_{\geq 0}$ for the acquisition function $a_N(\boldsymbol{x})$ in (4.20a); (vi) Safeguard threshold $\epsilon_{\Delta Y} \in \mathbb{R}_{>0}$ for ΔY in (4.20a).

Output: (i) Best cost obtained by the procedure $y_{best}(N_{max})$; (ii) Best sample obtained by the procedure $\boldsymbol{x_{best}}(N_{max})$.

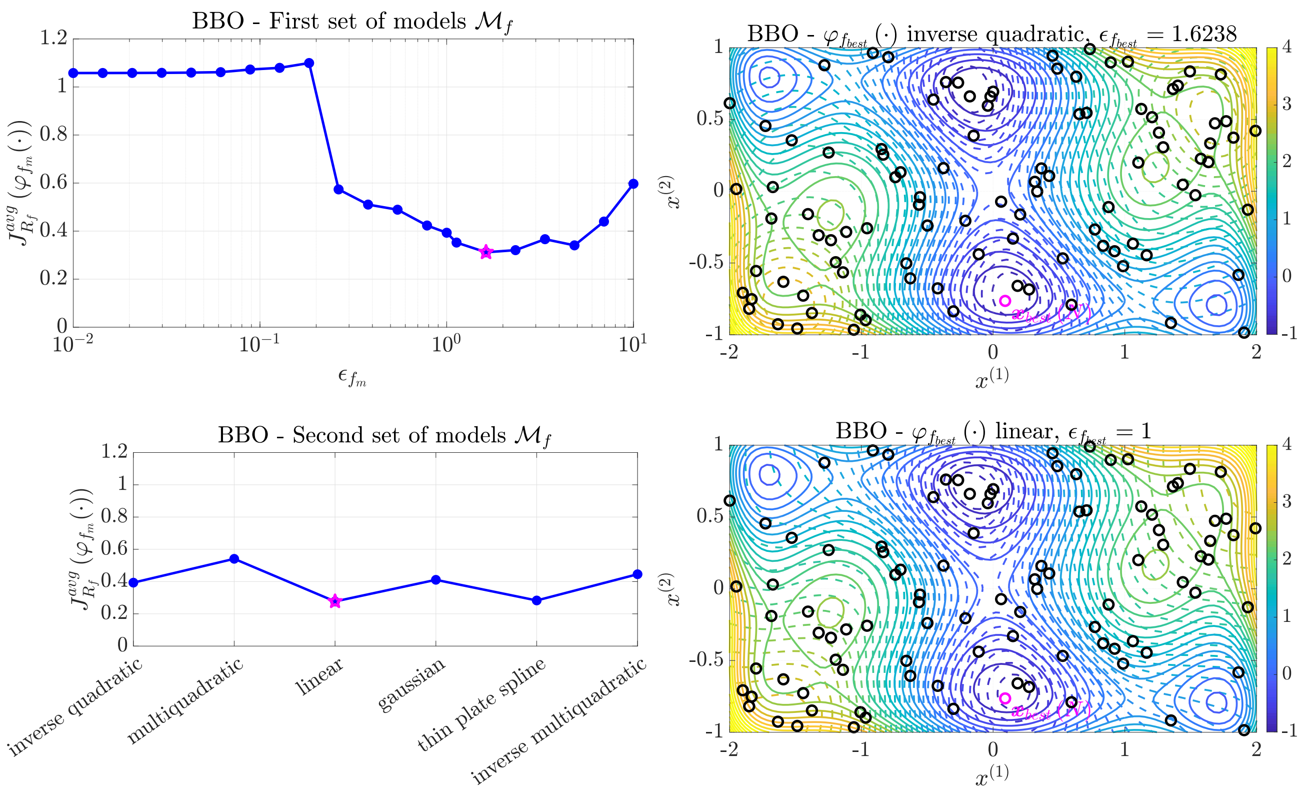
```
1: Rescale the GOP (2.1) as proposed in Section 4.4.1, obtaining Problem (4.27)
2: Generate a set $\mathcal{X}$ in (2.9) of $N_{init}$ starting points using a LHD (see Section 2.4)
3: Measure the black-box cost function $f(\cdot)$ at each $\boldsymbol{x}_i \in \mathcal{X}$, obtaining the set $\mathcal{Y}$ in (2.10)
4: Set $N = N_{init}$
5: for $k = 1, 2, \dots, N_{max} - N_{init}$ do
6:     Build the surrogate model for the cost function, $\hat{f}_N(\boldsymbol{x})$ in (4.1), from $\mathcal{X}$ and $\mathcal{Y}$ as in (4.4)
7:     Look for the next candidate sample $\boldsymbol{x}_{N+1}$ by solving Problem (4.23) with $a_N(\boldsymbol{x})$ in (4.20a)
8:     Measure the black-box cost function $f(\cdot)$ at $\boldsymbol{x}_{N+1}$, obtaining $y_{N+1}$
9:     Update the set of samples $\mathcal{X}$ and the set of measures $\mathcal{Y}$
10:    Check if $\boldsymbol{x}_{N+1}$ improves upon $\boldsymbol{x_{best}}(N)$, as highlighted by the flag $hasImproved$ from Algorithm 9
11:    if $hasImproved$ then set $\boldsymbol{x_{best}}(N+1) = \boldsymbol{x}_{N+1}$ else keep $\boldsymbol{x_{best}}(N+1) = \boldsymbol{x_{best}}(N)$
12:    Set $N = N + 1$
```

Algorithm 11: `GLISp` [11]

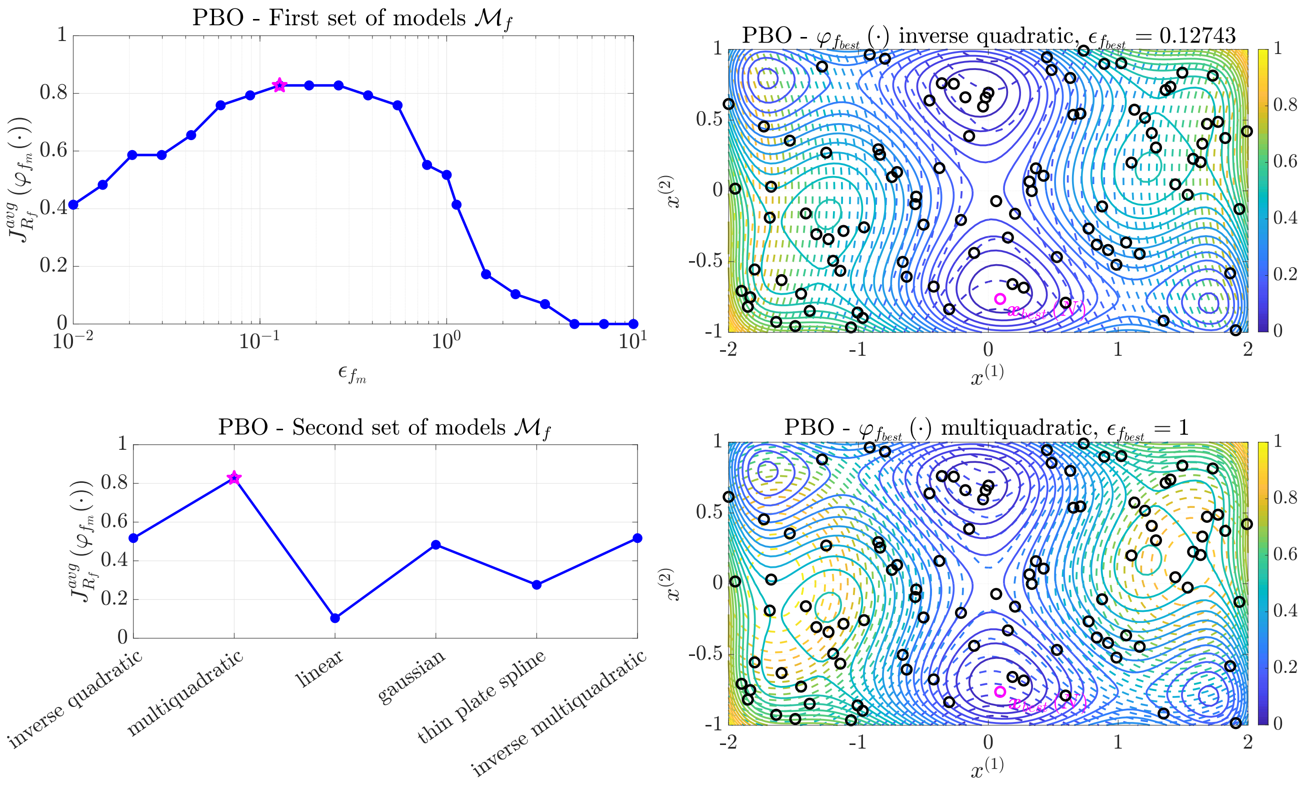
Input: (i) A-priori known constraint set Ω of the GOP (2.1); (ii) Number of initial samples $N_{init} \in \mathbb{N}$; (iii) Budget $N_{max} \in \mathbb{N}, N_{max} > N_{init}$; (iv) Hyper-parameters for the surrogate model $\hat{f}_N(\boldsymbol{x})$ in (4.1), namely shape parameter $\epsilon_f \in \mathbb{R}_{>0}$, radial function $\varphi_f(\cdot)$, regularization parameter $\lambda_f \in \mathbb{R}_{\geq 0}$ and tolerance $\sigma_\pi \in \mathbb{R}_{>0}$; (v) Exploration-exploitation trade-off weight $\delta \in \mathbb{R}_{\geq 0}$ for the acquisition function $a_N(\boldsymbol{x})$ in (4.20b); (vi) Set of models $\mathcal{M}_f$ in (4.30) for the recalibration of the surrogate model $\hat{f}_N(\boldsymbol{x})$ in (4.1); (vii) Set of indexes for the recalibration of the surrogate model $\hat{f}_N(\boldsymbol{x})$ in (4.1), $\mathcal{K}_{R_f} \subseteq \{1, \dots, N_{max} - N_{init}\}$, and folds ratio $R_f \in [0, 1]$.

Output: (i) Best sample obtained by the procedure $\boldsymbol{x_{best}}(N_{max})$.

```
1: Rescale the GOP (2.1) as proposed in Section 4.4.1, obtaining Problem (4.27)
2: Generate a set $\mathcal{X}$ in (2.9) of $N_{init}$ starting points using a LHD (see Section 2.4)
3: Evaluate the samples in $\mathcal{X}$ by querying the decision-maker as in Algorithm 7, obtaining the sets $\mathcal{B}$ (3.9)
   and $\mathcal{S}$ (3.10), as well as the best candidate $\boldsymbol{x_{best}}(N_{init})$
4: Set $N = N_{init}, M = |\mathcal{B}|$
5: for $k = 1, 2, \dots, N_{max} - N_{init}$ do
6:     if $k \in \mathcal{K}_{R_f}$ then recalibrate the surrogate model $\hat{f}_N(\boldsymbol{x})$ in (4.1) as in Algorithm 8
7:     Build the surrogate model for the scoring function, $\hat{f}_N(\boldsymbol{x})$ in (4.1), from $\mathcal{X}, \mathcal{B}$ and $\mathcal{S}$ by solving Problem
       (4.10)
8:     Look for the next candidate sample $\boldsymbol{x}_{N+1}$ by solving Problem (4.23) with $a_N(\boldsymbol{x})$ in (4.20b)
9:     Let the human decision-maker express a preference between $\boldsymbol{x}_{N+1}$ and $\boldsymbol{x_{best}}(N)$, obtaining $b_{M+1} =$
       $\pi_{\gtrsim}(\boldsymbol{x}_{N+1}, \boldsymbol{x_{best}}(N))$
10:    Update the set of samples $\mathcal{X}$, the set of preferences $\mathcal{B}$ and the mapping set $\mathcal{S}$
11:    Check if $\boldsymbol{x}_{N+1}$ improves upon $\boldsymbol{x_{best}}(N)$, as highlighted by the flag $hasImproved$ from Algorithm 9
12:    if $hasImproved$ then set $\boldsymbol{x_{best}}(N+1) = \boldsymbol{x}_{N+1}$ else keep $\boldsymbol{x_{best}}(N+1) = \boldsymbol{x_{best}}(N)$
13:    Set $N = N + 1, M = M + 1$
```



(a) Black-box optimization surrogates.



(b) Preference-based optimization surrogates.

Figure 18: Results of the recalibration procedure in Algorithm 8 applied as described in Example 4.6. On the left: performances of each model in $\mathcal{M}_f$, both for BBO and PBO; the star in magenta denotes the best one. On the right: comparison between the real function $f(x)$ (continuous line) and the surrogate $\hat{f}_N(x)$ (dashed line) with shape parameter and RBF obtained by the recalibration procedure. The circles denote all $N = 100$ available samples; the best candidate $x_{best}(N)$ is highlighted in magenta.

Algorithm 12: `C-GLISp` [156]

Input: (i) A-priori known constraint set Ω of the GOP (2.1); (ii) Number of initial samples $N_{init} \in \mathbb{N}$; (iii) Budget $N_{max} \in \mathbb{N}, N_{max} > N_{init}$; (iv) Hyper-parameters for the surrogate model $\hat{f}_N(\boldsymbol{x})$ in (4.1), namely shape parameter $\epsilon_f \in \mathbb{R}_{>0}$, radial function $\varphi_f(\cdot)$, regularization parameter $\lambda_f \in \mathbb{R}_{\geq 0}$ and tolerance $\sigma_\pi \in \mathbb{R}_{>0}$; (v) Exploration-exploitation trade-off weight $\delta \in \mathbb{R}_{\geq 0}$ and default penalty weight $\delta_{\Xi,default} \in \mathbb{R}_{\geq 0}$ for the acquisition function $a_N(\boldsymbol{x})$ in (4.20d); (vi) Set of models $\mathcal{M}_f$ in (4.30) for the recalibration of the surrogate model $\hat{f}_N(\boldsymbol{x})$ in (4.1); (vii) Set of indexes for the recalibration of the surrogate model $\hat{f}_N(\boldsymbol{x})$ in (4.1), $\mathcal{K}_{R_f} \subseteq \{1, \dots, N_{max} - N_{init}\}$, and folds ratio $R_f \in [0, 1]$.

Output: (i) Best sample obtained by the procedure $\boldsymbol{x_{best}}(N_{max})$.

```
1: Rescale the GOP (2.1) as proposed in Section 4.4.1, obtaining Problem (4.27)
2: Generate a set $\mathcal{X}$ in (2.9) of $N_{init}$ starting points using a LHD (see Section 2.4)
3: Evaluate the samples in $\mathcal{X}$ by querying the decision-maker as in Algorithm 7, obtaining the sets $\mathcal{B}$ (3.9)
   and $\mathcal{S}$ (3.10), as well as the best candidate $\boldsymbol{x_{best}}(N_{init})$
4: Evaluate the $\Xi$-feasibility of each $\boldsymbol{x}_i \in \mathcal{X}$, obtaining the set $\mathcal{U}_\Xi$ in (2.11)
5: Set $N = N_{init}, M = |\mathcal{B}|$
6: for $k = 1, 2, \dots, N_{max} - N_{init}$ do
7:     if $k \in \mathcal{K}_{R_f}$ then recalibrate the surrogate model $\hat{f}_N(\boldsymbol{x})$ in (4.1) as in Algorithm 8
8:     Build the surrogate model for the scoring function, $\hat{f}_N(\boldsymbol{x})$ in (4.1), from $\mathcal{X}, \mathcal{B}$ and $\mathcal{S}$ by solving Problem
       (4.10)
9:     Build the surrogate model for the black-box constraints functions in $\Xi$, i.e. estimate the probability of
       $\Xi$-feasibility $p_N(\boldsymbol{x} \in \Xi)$ as in (4.16) from $\mathcal{X}$ and $\mathcal{U}_\Xi$
10:    Recalibrate $\delta_\Xi$ as in (4.22)
11:    Look for the next candidate sample $\boldsymbol{x}_{N+1}$ by solving Problem (4.23) with $a_N(\boldsymbol{x})$ in (4.20d)
12:    Let the human decision-maker express a preference between $\boldsymbol{x}_{N+1}$ and $\boldsymbol{x_{best}}(N)$, obtaining $b_{M+1} =$
       $\pi_{\gtrsim}(\boldsymbol{x}_{N+1}, \boldsymbol{x_{best}}(N))$
13:    Evaluate the $\Xi$-feasiblity of $\boldsymbol{x}_{N+1}$, obtaining $u_{N+1} = u_\Xi(\boldsymbol{x}_{N+1})$
14:    Update the sets $\mathcal{X}, \mathcal{B}, \mathcal{S}$ and $\mathcal{U}_\Xi$
15:    Check if $\boldsymbol{x}_{N+1}$ improves upon $\boldsymbol{x_{best}}(N)$, as highlighted by the flag $hasImproved$ from Algorithm 9
16:    if $hasImproved$ then set $\boldsymbol{x_{best}}(N+1) = \boldsymbol{x}_{N+1}$ else keep $\boldsymbol{x_{best}}(N+1) = \boldsymbol{x_{best}}(N)$
17:    Set $N = N + 1, M = M + 1$
```

Part II

Theoretical contributions

Chapter 5. Globally convergent extensions of `GLIS` and `GLISp`

This Chapter presents two contributions of this book in the context of unconstrained black-box and preference-based optimization. The former is the extension of the `GLIS` [10] and `GLISp` [11] procedures, while the latter is the derivation of a unified surrogate-based scheme that can be employed both for BBO and PBO. *We argue that black-box and preference-based optimization should be treated in a unified fashion*, the only difference being the definition of the surrogate model $\hat{f}_N(\boldsymbol{x})$, obtained either from the measures of $f(\boldsymbol{x})$ or the preferences expressed by the decision-maker. Hence, many of the strategies followed by black-box optimization methods (in Chapter 2), such as cycling the exploration-exploitation trade-off weights, can also be used for PBO. Furthermore, *we can even address the convergence of preference-based optimization algorithms by leveraging the results in Section 3.1.* Notably, to the best of our knowledge, no proofs of convergence are available for any of the surrogate-based PBO methods reviewed in Chapter 3. Instead, in this book, we will prove the global convergence of all the algorithms proposed in this Chapter.

The rest of this Chapter is organized as follows. Section 5.1 thoroughly analyzes the IDW distance function in (4.17), highlighting its shortcomings when it comes to the exploration of the feasible region Ω of the GOP (2.1). Then, in Section 5.2, we propose a revisited infill sampling criterion that addresses the limitations of $z_N(\boldsymbol{x})$ in (4.17). Section 5.3 reports our extensions of the `GLIS` [10] and `GLISp` [11] procedures, which we refer to as `GLIS-r` [108] and `GLISp-r` [109] respectively. We also address their convergence to the global minima of the GOP (2.1). In Section 5.4, we describe a globally convergent unified surrogate-based scheme for black-box and preference based optimization. We refer to the latter as generalized Metric Response Surface (`gMRS` [108]), since it is closely related to the popular `MSRS` [116] scheme used for black-box optimization and from which `MSRBF` [116] originates (see Section 2.6). Finally, Section 5.5 summarizes the results presented in this Chapter.

5.1 In-depth analysis of the IDW distance function

In this Section, we analyze the IDW distance function $z_N(\boldsymbol{x})$ in (4.17) in great detail. In short, we have empirically observed that the exploratory capabilities of $z_N(\boldsymbol{x})$ diminish rapidly as the number of samples, N, increases. Furthermore, the scaling constants ΔY and $\Delta \hat{F}$ in (4.20a) and (4.20b) are often not sufficient to make the surrogate model $\hat{f}_N(\boldsymbol{x})$ in (4.1) comparable to the latter exploration function. This, in turn, can result in `GLIS` [10] and `GLISp` [11] missing the global minima of the GOP (2.1). In practice, `GLIS` [10] is less affected by the shortcomings of $z_N(\boldsymbol{x})$ in (4.17), since it also includes an additional exploration function, namely the IDW variance function in (4.19). Instead, as we will see in Chapter 7, GLISp [11] is more prone to getting stuck on local minima of $f(\boldsymbol{x})$ compared to its BBO counterpart. This is due to the fact that the acquisition function in (4.20b) relies heavily on $z_N(\boldsymbol{x})$ in (4.17).

5.1.1 Shortcomings of the IDW distance function

There are two main shortcomings of $z_N(\boldsymbol{x})$ in (4.17) that can make its contribution negligible in $a_N(\boldsymbol{x})$ in (4.20a) and (4.20b) and complicate the selection of the trade-off weights δ_1 (GLIS [10]) and δ (GLISp [11]):

1. Even though the range of $z_N(\boldsymbol{x})$ is $(-1, 0]$ (see Definition 4.3), what we are really interested in when solving Problem (4.23) and, ultimately, the GOP (2.1), are the values that it assumes for $\boldsymbol{x} \in \Omega$ and not on its whole domain $\mathbb{R}^n$. In particular, *there are some situations for which* $|z_N(\boldsymbol{x})| \ll 1, \forall \boldsymbol{x} \in \Omega$, *making its contribution in the acquisition functions* (4.20a) *and* (4.20b) *negligible*. Consider, for example, the case $\mathcal{X} = \{\boldsymbol{x}_1\}$ ($N = 1$); then, $\forall \boldsymbol{x} \in \mathbb{R}^n \setminus \mathcal{X}$, the IDW distance function simply becomes:

$$z_1(\boldsymbol{x}) = -\frac{2}{\pi} \cdot \arctan\left(\|\boldsymbol{x} - \boldsymbol{x}_1\|_2^2\right).$$

   Suppose that the GOP (2.1) is only bound constrained, i.e. $\Omega = \{\boldsymbol{x} : \boldsymbol{l} \leq \boldsymbol{x} \leq \boldsymbol{u}\}$. Then, $z_1(\boldsymbol{x})$ assumes its lowest value at one (or more) of the vertices of the box defined by the constraints in Ω. Define $\boldsymbol{v}_\Omega \in \Omega$ as one of such vertices, if $\|\boldsymbol{v}_\Omega - \boldsymbol{x}_1\|_2$ is close to zero, then $|z_1(\boldsymbol{x})| \ll 1, \forall \boldsymbol{x} \in \Omega$. Thus, unless $\delta_1 \gg 1$ in (4.20a) or $\delta \gg 1$ in (4.20b), $\hat{f}_N(\boldsymbol{x})$ and $z_N(\boldsymbol{x})$ might not be comparable.

2. *The (absolute) values assumed by the IDW distance function tend to decrease as the number of samples increases*. To clarify this, consider two sets of samples:

$$\mathcal{X}' = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}, \quad |\mathcal{X}'| = N,$$
$$\mathcal{X}'' = \mathcal{X}' \cup \{\boldsymbol{x}_{N+1}\}, \quad |\mathcal{X}''| = N + 1.$$

   Given any point $\tilde{\boldsymbol{x}} \in \mathbb{R}^n \setminus \mathcal{X}''$, the IDW distance functions obtained from the previously defined sets are:

$$z_N(\tilde{\boldsymbol{x}}) = -\frac{2}{\pi} \cdot \arctan\left(\frac{1}{\sum_{i=1}^{N} w_i(\tilde{\boldsymbol{x}})}\right),$$
$$z_{N+1}(\tilde{\boldsymbol{x}}) = -\frac{2}{\pi} \cdot \arctan\left(\frac{1}{\sum_{i=1}^{N+1} w_i(\tilde{\boldsymbol{x}})}\right).$$

Note that $w_i(\tilde{\boldsymbol{x}}) > 0, \forall \tilde{\boldsymbol{x}} \in \mathbb{R}^n \setminus \mathcal{X}''$ and $i = 1, \ldots, N+1$ (see Definition 4.1). Hence:

$$|z_N(\tilde{\boldsymbol{x}})| > |z_{N+1}(\tilde{\boldsymbol{x}})| > 0,$$

proving the above point. In practice this means that, unless δ_1 in (4.20a) and δ in (4.20b) are progressively increased as the iterations go on, GLIS [10] and (mostly) GLISp [11] will explore the feasible set Ω of the GOP (2.1) less as the number of samples increases, regardless of whether a region which contains the global minimizer(s) in $\mathcal{X}^*$ (2.3) has been located.

A visualization of these two shortcomings is presented in Figure 19.

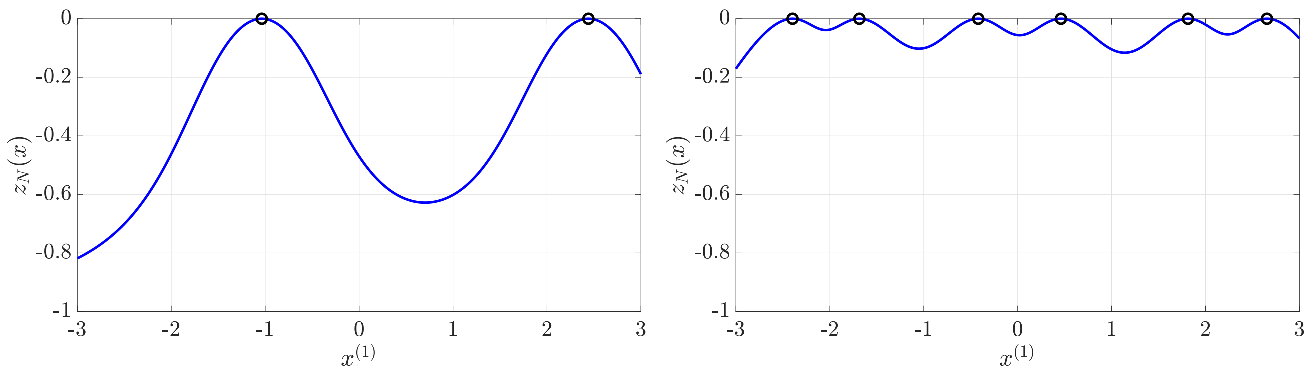


Figure 19: Examples of the IDW distance function $z_N(x)$ in (4.17) for different numbers of points ($N = 2$ on the left, $N = 6$ on the right) and $-3 = l \le x \le u = 3$. Notice how $z_N(x)$ does not cover its whole range $(-1, 0]$, at least inside the domain imposed by the bound constraints, and its absolute values decrease as the number of samples increases.

The limitations of $z_N(\boldsymbol{x})$ in (4.17) also highlight how a simple acquisition function defined as:

$$a_N(\boldsymbol{x}) = \delta \cdot \hat{f}_N(\boldsymbol{x}) + (1 - \delta) \cdot z_N(\boldsymbol{x}),$$

where $\delta \in [0, 1]$ is a exploration-exploitation trade-off weight, is not suitable for the search of the next candidate sample as in Problem (4.23). That is because: (i) $\hat{f}_N(\boldsymbol{x})$ in (4.1) and $z_N(\boldsymbol{x})$ in (4.17) likely assume different ranges and (ii) the absolute values assumed by the IDW distance function tend to decrease as N increases. For this reason, in the acquisition functions used by GLIS [10] and GLISp [11], the contributions are rescaled by two constants, $\Delta Y = \max\left\{\max_{y_i \in \mathcal{Y}} y_i - \min_{y_i \in \mathcal{Y}} y_i, \epsilon_{\Delta Y}\right\}$ and $\Delta \hat{F} = \max_{\boldsymbol{x}_i \in \mathcal{X}} \hat{f}_N(\boldsymbol{x}_i) - \min_{\boldsymbol{x}_i \in \mathcal{X}} \hat{f}_N(\boldsymbol{x}_i)$ (see (4.20a) and (4.20b)). However, as we will see in Section 5.2, it might still not be enough to make $\hat{f}_N(\boldsymbol{x})$ in (4.1) and $z_N(\boldsymbol{x})$ in (4.17) assume similar ranges.
In this book, we propose to *rescale the surrogate model and the IDW distance function through min-max rescaling, using some insights on the stationary points of $z_N(\boldsymbol{x})$ in* (4.17). Min-max rescaling has also been successfully applied in algorithm MSRS [116] to make the contributions of $\hat{f}_N(\boldsymbol{x})$ in (2.29) and $z_N(\boldsymbol{x})$ in (2.51) comparable (see the acquisition function $a_N(\boldsymbol{x})$ in (2.52)), although the samples used for the rescaling are generated randomly, regardless of $\hat{f}_N(\boldsymbol{x})$ and $z_N(\boldsymbol{x})$.

5.1.2 Stationary points of the IDW distance function

In order to gain some insights on the stationary points of $z_N(\boldsymbol{x})$ in (4.17), we first compute its gradient. Recall that the IDW distance function is differentiable everywhere, see Proposition 4.4. The following Lemma reports its gradient.

Lemma 5.1: Gradient of the IDW distance function. *The gradient of $z_N(\boldsymbol{x})$ in* (4.17) *is:*

$$\nabla_{\boldsymbol{x}} z_N(\boldsymbol{x}) = \begin{cases} \mathbf{0}_n & \text{if } \boldsymbol{x} \in \mathcal{X} \\ -\frac{4}{\pi} \cdot \frac{\sum_{i=1}^N (\boldsymbol{x}-\boldsymbol{x}_i)\cdot w_i(\boldsymbol{x})^2}{1+\left[\sum_{i=1}^N w_i(\boldsymbol{x})\right]^2} & \text{otherwise} \end{cases}. \tag{5.1}$$

Proof. Consider the case $\boldsymbol{x} \in \mathbb{R}^n \setminus \mathcal{X}$. We compute the gradient of the IDW distance function in (4.17) by repeatedly applying the chain rule. Recall that the gradient of the squared Euclidean norm is (see (2.24)):

$$\nabla_{\boldsymbol{x}} \|\boldsymbol{x}-\boldsymbol{x}_i\|_2^2 = 2\cdot(\boldsymbol{x}-\boldsymbol{x}_i), \quad \forall \boldsymbol{x} \in \mathbb{R}^n.$$

Consider now the IDW function $w_i(\boldsymbol{x})$ in (4.11), which is differentiable $\forall \boldsymbol{x} \in \mathbb{R}^n \setminus \{\boldsymbol{x}_i\}$ (see Proposition 4.2). We can easily compute its gradient:

$$\begin{aligned}\nabla_{\boldsymbol{x}} w_i(\boldsymbol{x}) &= \left.\frac{d}{dt}\frac{1}{t}\right|_{t=\|\boldsymbol{x}-\boldsymbol{x}_i\|_2^2} \cdot \nabla_{\boldsymbol{x}} \|\boldsymbol{x}-\boldsymbol{x}_i\|_2^2 \\ &= -\left.\frac{1}{t^2}\right|_{t=\|\boldsymbol{x}-\boldsymbol{x}_i\|_2^2} \cdot 2\cdot(\boldsymbol{x}-\boldsymbol{x}_i) \\ &= -2\cdot\frac{\boldsymbol{x}-\boldsymbol{x}_i}{\|\boldsymbol{x}-\boldsymbol{x}_i\|_2^4} \\ &= -2\cdot(\boldsymbol{x}-\boldsymbol{x}_i)\cdot w_i(\boldsymbol{x})^2, \quad \forall \boldsymbol{x} \in \mathbb{R}^n \setminus \{\boldsymbol{x}_i\}.\end{aligned}$$

Now, let us focus on the argument of the $\arctan(\cdot)$ function in $z_N(\boldsymbol{x})$, which is (see (4.17)):

$$h(\boldsymbol{x}) = \frac{1}{\sum_{i=1}^N w_i(\boldsymbol{x})}, \quad \forall \boldsymbol{x} \in \mathbb{R}^n \setminus \mathcal{X}.$$

We have that:

$$\begin{aligned}\nabla_{\boldsymbol{x}} h(\boldsymbol{x}) &= \left.\frac{d}{dt}\frac{1}{t}\right|_{t=\sum_{i=1}^N w_i(\boldsymbol{x})} \cdot \sum_{i=1}^N \nabla_{\boldsymbol{x}} w_i(\boldsymbol{x}) \\ &= -\left.\frac{1}{t^2}\right|_{t=\sum_{i=1}^N w_i(\boldsymbol{x})} \cdot \left[-2\cdot\sum_{i=1}^N (\boldsymbol{x}-\boldsymbol{x}_i)\cdot w_i(\boldsymbol{x})^2\right] \\ &= 2\cdot\frac{\sum_{i=1}^N (\boldsymbol{x}-\boldsymbol{x}_i)\cdot w_i(\boldsymbol{x})^2}{\left[\sum_{i=1}^N w_i(\boldsymbol{x})\right]^2}, \quad \forall \boldsymbol{x} \in \mathbb{R}^n \setminus \mathcal{X}.\end{aligned}$$

Finally, using the chain rule one last time, we can compute the gradient of $z_N(\boldsymbol{x})$ in (4.17) for all $\boldsymbol{x} \in \mathbb{R}^n \setminus \mathcal{X}$:

$$\begin{aligned}
\nabla_{\boldsymbol{x}} z_N(\boldsymbol{x}) &= \frac{d}{dt}\left[-\frac{2}{\pi}\cdot\arctan(t)\right]\Bigg|_{t=h(\boldsymbol{x})}\cdot\nabla_{\boldsymbol{x}} h(\boldsymbol{x}) \\
&= -\frac{2}{\pi}\cdot\frac{1}{1+t^2}\Bigg|_{t=h(\boldsymbol{x})}\cdot 2\cdot\frac{\sum_{i=1}^N(\boldsymbol{x}-\boldsymbol{x}_i)\cdot w_i(\boldsymbol{x})^2}{\left[\sum_{i=1}^N w_i(\boldsymbol{x})\right]^2} \\
&= -\frac{4}{\pi}\cdot\frac{1}{1+\left[\frac{1}{\sum_{i=1}^N w_i(\boldsymbol{x})}\right]^2}\cdot\frac{\sum_{i=1}^N(\boldsymbol{x}-\boldsymbol{x}_i)\cdot w_i(\boldsymbol{x})^2}{\left[\sum_{i=1}^N w_i(\boldsymbol{x})\right]^2} \\
&= -\frac{4}{\pi}\cdot\frac{\cancel{\left[\sum_{i=1}^N w_i(\boldsymbol{x})\right]^2}}{1+\left[\sum_{i=1}^N w_i(\boldsymbol{x})\right]^2}\cdot\frac{\sum_{i=1}^N(\boldsymbol{x}-\boldsymbol{x}_i)\cdot w_i(\boldsymbol{x})^2}{\cancel{\left[\sum_{i=1}^N w_i(\boldsymbol{x})\right]^2}} \\
&= -\frac{4}{\pi}\cdot\frac{\sum_{i=1}^N(\boldsymbol{x}-\boldsymbol{x}_i)\cdot w_i(\boldsymbol{x})^2}{1+\left[\sum_{i=1}^N w_i(\boldsymbol{x})\right]^2}, \quad \forall \boldsymbol{x}\in\mathbb{R}^n\setminus\mathcal{X}. \qquad (5.2)
\end{aligned}$$

Now, let us consider the case $\boldsymbol{x}_i \in \mathcal{X}$. In the Proof of Proposition 4.4, we have shown that the partial derivatives of $z_N(\boldsymbol{x})$ in (4.17) are zero at each $\boldsymbol{x}_i \in \mathcal{X}$, i.e.

$$\nabla_{\boldsymbol{x}} z_N(\boldsymbol{x}_i) = \boldsymbol{0}_n, \quad \forall \boldsymbol{x}_i \in \mathcal{X}. \qquad (5.3)$$

Lastly, combining (5.2) and (5.3), we obtain the expression for the gradient of the IDW distance function $\forall \boldsymbol{x} \in \mathbb{R}^n$, as reported in (5.1). □

The locations of the global maximizers of $z_N(\boldsymbol{x})$ in (4.17) can be deduced immediately from Definition 4.3 and Lemma 5.1.

Proposition 5.1: Global maximizers of the IDW distance function. *Each $\boldsymbol{x}_i \in \mathcal{X}$ in* (2.9) *is a global maximizer of $z_N(\boldsymbol{x})$ in* (4.17).

Proof. Recall that:

(i) $\nabla_{\boldsymbol{x}} z_N(\boldsymbol{x}_i) = \boldsymbol{0}_n, \forall \boldsymbol{x}_i \in \mathcal{X}$, see (5.1);

(ii) $z_N(\boldsymbol{x}) < 0, \forall \boldsymbol{x} \in \mathbb{R}^n \setminus \mathcal{X}$, see (4.17);

(iii) $z_N(\boldsymbol{x}_i) = 0, \forall \boldsymbol{x}_i \in \mathcal{X}$, see (4.17).

From Item (i) we deduce that each $\boldsymbol{x}_i \in \mathcal{X}$ is a stationary point of $z_N(\boldsymbol{x})$. Item (ii), in conjunction with Item (iii), implies that such samples are local maximizers of the IDW distance function in (4.17) since there exists a neighborhood of $\boldsymbol{x}_i \in \mathcal{X}$, $\mathcal{N}(\boldsymbol{x}_i)$, such that $z_N(\boldsymbol{x}) \leq z_N(\boldsymbol{x}_i), \forall \boldsymbol{x} \in \mathcal{N}(\boldsymbol{x}_i)$. Moreover, due to Item (ii), $z_N(\boldsymbol{x}) \leq z_N(\boldsymbol{x}_i), \forall \boldsymbol{x} \in \mathbb{R}^n$ and not just in a neighborhood of $\boldsymbol{x}_i \in \mathcal{X}$. Hence, each $\boldsymbol{x}_i \in \mathcal{X}$ is a global maximizer of $z_N(\boldsymbol{x})$ in (4.17). □

Reaching similar conclusions for the minimizers of $z_N(\boldsymbol{x})$ in (4.17) is much harder; however, we can consider some simplified situations. Note that *we are not necessarily interested in finding the minimizers of the IDW distance function in* (4.17) *with high accuracy, but rather to gain some insights on where they are likely to be located so that we can rescale both $z_N(\boldsymbol{x})$ in* (4.17) *and $\hat{f}_N(\boldsymbol{x})$ in* (4.1) *sufficiently enough to make them comparable*. Moreover, their approximate locations can be used to solve the following optimization problem (*pure exploration*):

$$\begin{aligned} \boldsymbol{x}_{N+1} = \arg\min_{\boldsymbol{x}} & \; z_N(\boldsymbol{x}) \\ \text{s.t.} \quad & \boldsymbol{x} \in \Omega \end{aligned} \tag{5.4}$$

by using a *multi-start derivative-based optimization method* (see Section 1.2.4). Problem (5.4) is quite relevant for the convergence of the algorithms that we will propose in Section 5.3.

Remark 5.1. *In the following Paragraphs, we analyze where the local minimizers of $z_N(\boldsymbol{x})$ in* (4.17)*, as well as the solution(s) of the simplified problem:*

$$\begin{aligned} \boldsymbol{x}_{N+1} = \arg\min_{\boldsymbol{x}} & \; z_N(\boldsymbol{x}) \\ \textit{s.t.} \quad & \boldsymbol{l} \leq \boldsymbol{x} \leq \boldsymbol{u}, \end{aligned} \tag{5.5}$$

are located in some specific cases. In practice, since $\{\boldsymbol{x} : \boldsymbol{l} \leq \boldsymbol{x} \leq \boldsymbol{u}\} \supseteq \Omega$, the global minimum $z_N(\boldsymbol{x}_{N+1})$ of Problem (5.5) *is lower or at most equal to the global minimum of Problem* (5.4)*. Therefore, the minimizers of Problem* (5.5) *are better suited to perform min-max rescaling of $z_N(\boldsymbol{x})$ in* (4.17) *than those of Problem* (5.4).

Case $\mathcal{X} = \{\boldsymbol{x}_1\}$ ($N = 1$). The IDW distance function and its gradient $\forall \boldsymbol{x} \in \mathbb{R}^n \setminus \mathcal{X}$ are:

$$\begin{aligned} z_N(\boldsymbol{x}) &= -\frac{2}{\pi} \cdot \arctan\left(\|\boldsymbol{x} - \boldsymbol{x}_1\|_2^2\right), \\ \nabla_{\boldsymbol{x}} z_N(\boldsymbol{x}) &= -\frac{4}{\pi} \cdot (\boldsymbol{x} - \boldsymbol{x}_1) \cdot \frac{w_1(\boldsymbol{x})^2}{1 + w_1(\boldsymbol{x})^2}. \end{aligned}$$

Clearly, $\forall \boldsymbol{x} \in \mathbb{R}^n \setminus \mathcal{X}$, the gradient is never zero since $w_1(\boldsymbol{x}) > 0$. Therefore, the only stationary point is the global maximizer $\boldsymbol{x}_1 \in \mathcal{X}$ (see Proposition 5.1). However, if we were to consider Problem (5.5), then its solution would be located at one of the vertices of the box defined by the bound constraints $\boldsymbol{l} \leq \boldsymbol{x} \leq \boldsymbol{u}$.

Case $\mathcal{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2\}$ ($N = 2$). The gradient of the IDW distance function $\forall \boldsymbol{x} \in \mathbb{R}^n \setminus \mathcal{X}$ is:

$$\nabla_{\boldsymbol{x}} z_N(\boldsymbol{x}) = -\frac{4}{\pi} \cdot \frac{(\boldsymbol{x} - \boldsymbol{x}_1) \cdot w_1(\boldsymbol{x})^2 + (\boldsymbol{x} - \boldsymbol{x}_2) \cdot w_2(\boldsymbol{x})^2}{1 + [w_1(\boldsymbol{x}) + w_2(\boldsymbol{x})]^2}.$$

We calculate the stationary points of $z_N(\boldsymbol{x})$ in (4.17) by setting the gradient equal to zero:

$$\begin{aligned}
\nabla_{\boldsymbol{x}} z_N(\boldsymbol{x}) &= \boldsymbol{0}_n \\
-\frac{4}{\pi} \cdot \frac{(\boldsymbol{x} - \boldsymbol{x}_1) \cdot w_1(\boldsymbol{x})^2 + (\boldsymbol{x} - \boldsymbol{x}_2) \cdot w_2(\boldsymbol{x})^2}{1 + \underbrace{[w_1(\boldsymbol{x}) + w_2(\boldsymbol{x})]^2}_{>0, \forall \boldsymbol{x} \in \mathbb{R}^n \setminus \mathcal{X}}} &= \boldsymbol{0}_n \\
(\boldsymbol{x} - \boldsymbol{x}_1) \cdot w_1(\boldsymbol{x})^2 + (\boldsymbol{x} - \boldsymbol{x}_2) \cdot w_2(\boldsymbol{x})^2 &= \boldsymbol{0}_n.
\end{aligned}$$

Let us consider the *midpoint* $\boldsymbol{x}_\mu = \frac{\boldsymbol{x}_1 + \boldsymbol{x}_2}{2}$, that is such that $\left\|\boldsymbol{x}_\mu - \boldsymbol{x}_1\right\|_2 = \left\|\boldsymbol{x}_\mu - \boldsymbol{x}_2\right\|_2$ and for which $w_1(\boldsymbol{x}_\mu) = w_2(\boldsymbol{x}_\mu)$. If we substitute it in the previous expression, we obtain:

$$\begin{aligned}
(\boldsymbol{x}_\mu - \boldsymbol{x}_1) \cdot w_1(\boldsymbol{x}_\mu)^2 + (\boldsymbol{x}_\mu - \boldsymbol{x}_2) \cdot \overbrace{w_2(\boldsymbol{x}_\mu)^2}^{=w_1(\boldsymbol{x}_\mu)^2} &= \boldsymbol{0}_n \\
w_1(\boldsymbol{x}_\mu)^2 \cdot (\boldsymbol{x}_\mu - \boldsymbol{x}_1 + \boldsymbol{x}_\mu - \boldsymbol{x}_2) &= \boldsymbol{0}_n \\
2 \cdot \boldsymbol{x}_\mu - \boldsymbol{x}_1 - \boldsymbol{x}_2 &= \boldsymbol{0}_n \\
\cancel{2} \cdot \frac{\boldsymbol{x}_1 + \boldsymbol{x}_2}{\cancel{2}} - \boldsymbol{x}_1 - \boldsymbol{x}_2 &= \boldsymbol{0}_n \\
\boldsymbol{0}_n &= \boldsymbol{0}_n.
\end{aligned}$$

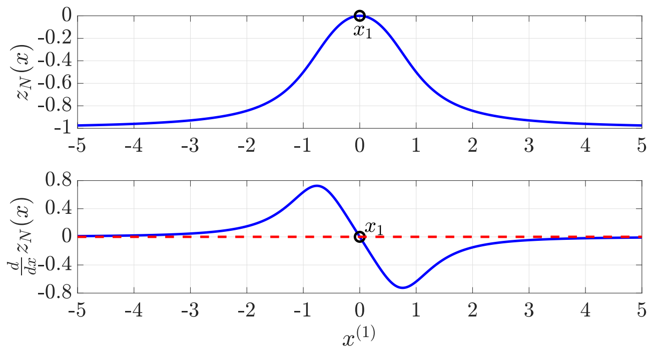
Hence,

$$\nabla_{\boldsymbol{x}} z_N(\boldsymbol{x}_\mu) = \boldsymbol{0}_n,$$

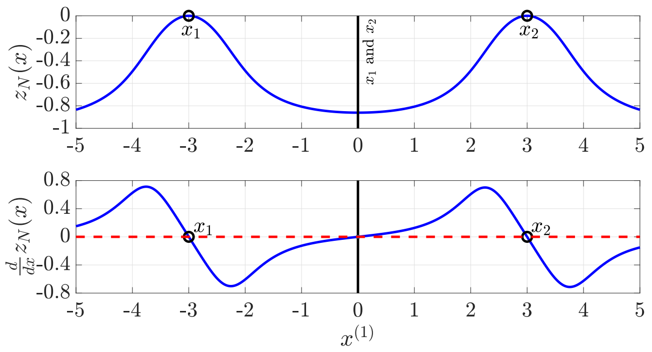
which means that $\boldsymbol{x}_\mu$ is a stationary point for $z_N(\boldsymbol{x})$ in (4.17). It is easy to see by visual inspection that such point is actually a local minimizer for the IDW distance function (see for example Figure 20). However, note that $\boldsymbol{x}_\mu$ is not necessarily the global solution of Problem (5.5), it might just be a local one.

Case $\mathcal{X} = \mathcal{X}^{(1)} \cup \mathcal{X}^{(2)}$ ($N > 2$). Suppose now that the samples contained in $\mathcal{X}$ (2.9) can be partitioned into two clusters:

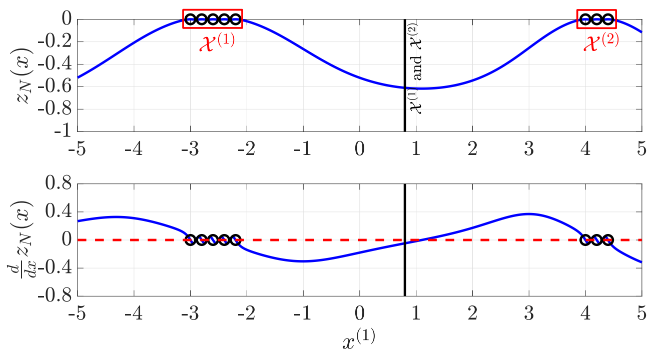
- $\mathcal{X}^{(1)} = \left\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{N_1}\right\}$ ($\left|\mathcal{X}^{(1)}\right| = N_1$),
- $\mathcal{X}^{(2)} = \left\{\boldsymbol{x}_{N_1+1}, \ldots, \boldsymbol{x}_N\right\}$ ($\left|\mathcal{X}^{(2)}\right| = N - N_1$),



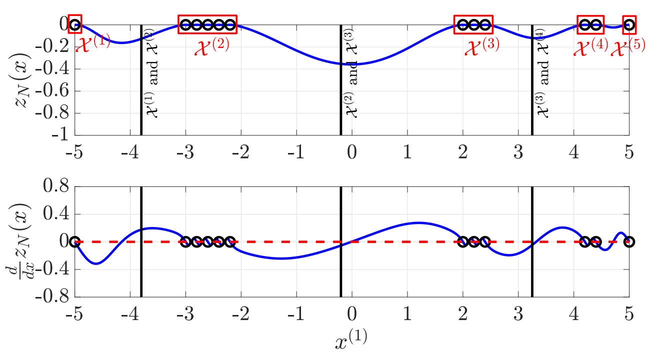
(a) $\mathcal{X} = \{x_1\}$.



(b) $\mathcal{X} = \{x_1, x_2\}$.



(c) $\mathcal{X} = \mathcal{X}^{(1)} \cup \mathcal{X}^{(2)}$.



(d) $\mathcal{X} = \mathcal{X}^{(1)} \cup \mathcal{X}^{(2)} \cup \mathcal{X}^{(3)} \cup \mathcal{X}^{(4)} \cup \mathcal{X}^{(5)}$.

Figure 20: One-dimensional examples of the IDW distance function $z_N(x)$ in (4.17) and its gradient $\nabla_x z_N(x)$ in (5.1) in the four analyzed cases. The red boxes mark the different clusters while the red dashed line highlights the values of x for which the first derivative of $z_N(x)$ is zero. Finally, the black vertical lines mark the midpoints (either between points or centroids of the clusters). Only a portion of all possible midpoints between centroids has been reported in the general case. Notice that the midpoints in Figure 20c and Figure 20d are quite close to the local minimizers of $z_N(x)$, while the midpoint in Figure 20b is exactly a local solution of Problem (5.5).

such that $\mathcal{X}^{(1)} \cap \mathcal{X}^{(2)} = \emptyset$ and $\mathcal{X}^{(1)} \cup \mathcal{X}^{(2)} = \mathcal{X}$. Consider the midpoint between the centroids of each cluster:

$$\boldsymbol{x}_\mu = \frac{1}{2} \cdot \left[\frac{\sum_{\boldsymbol{x}_i \in \mathcal{X}^{(1)}} \boldsymbol{x}_i}{N_1} + \frac{\sum_{\boldsymbol{x}_i \in \mathcal{X}^{(2)}} \boldsymbol{x}_i}{N - N_1} \right]. \tag{5.6}$$

We make the simplifying assumption that all the points contained inside each cluster are quite close to each other, i.e.

$$\left\| \boldsymbol{x}_i - \boldsymbol{x}_j \right\|_2 \approx 0, \quad \forall \boldsymbol{x}_i, \boldsymbol{x}_j \in \mathcal{X}^{(1)}, i \neq j, \tag{5.7}$$

and similarly for the points in $\mathcal{X}^{(2)}$. In turn, we have that $\boldsymbol{x}_1 \approx \boldsymbol{x}_2 \approx \ldots \approx \boldsymbol{x}_{N_1}$ and $\boldsymbol{x}_{N_1+1} \approx \boldsymbol{x}_{N_1+2} \approx \ldots \approx \boldsymbol{x}_N$. Then, the midpoint in (5.6) is approximately equal to:

$$\begin{aligned}
\boldsymbol{x}_\mu &= \frac{\frac{\sum_{\boldsymbol{x}_i \in \mathcal{X}^{(1)}} \boldsymbol{x}_i}{N_1} + \frac{\sum_{\boldsymbol{x}_i \in \mathcal{X}^{(2)}} \boldsymbol{x}_i}{N - N_1}}{2} \\
&\approx \frac{\frac{\cancel{N_1} \cdot \boldsymbol{x}_1}{\cancel{N_1}} + \frac{\cancel{(N - N_1)} \cdot \boldsymbol{x}_N}{\cancel{N - N_1}}}{2} \\
&= \frac{\boldsymbol{x}_1 + \boldsymbol{x}_N}{2}.
\end{aligned}$$

Now, consider any point $\boldsymbol{x}_{j_1} \in \mathcal{X}^{(1)}$ and $\boldsymbol{x}_{j_2} \in \mathcal{X}^{(2)}$. We have that:

$$\begin{aligned}
\left\|\boldsymbol{x}_\mu - \boldsymbol{x}_{j_1}\right\|_2 &\approx \left\|\frac{\boldsymbol{x}_1 + \boldsymbol{x}_N}{2} - \boldsymbol{x}_{j_1}\right\|_2 \\
&\qquad \text{Using the assumption } \boldsymbol{x}_1 \approx \ldots \approx \boldsymbol{x}_{j_1} \approx \ldots \approx \boldsymbol{x}_{N_1} \\
&= \left\|\frac{\boldsymbol{x}_1 + \boldsymbol{x}_N}{2} - \boldsymbol{x}_1\right\|_2 \\
&= \left\|\frac{\boldsymbol{x}_N - \boldsymbol{x}_1}{2}\right\|_2 .
\end{aligned}$$

Similarly,

$$\left\|\boldsymbol{x}_\mu - \boldsymbol{x}_{j_2}\right\|_2 \approx \left\|\frac{\boldsymbol{x}_1 - \boldsymbol{x}_N}{2}\right\|_2 ,$$

and thus by applying the properties of any norm (see Appendix A.2.1, Definition A.10):

$$\left\|\boldsymbol{x}_\mu - \boldsymbol{x}_{j_1}\right\|_2 \approx \left\|\boldsymbol{x}_\mu - \boldsymbol{x}_{j_2}\right\|_2 \approx \frac{1}{2} \cdot \|\boldsymbol{x}_N - \boldsymbol{x}_1\|_2 , \quad \forall \boldsymbol{x}_{j_1} \in \mathcal{X}^{(1)}, \boldsymbol{x}_{j_2} \in \mathcal{X}^{(2)}.$$

Therefore, all the IDW functions $w_i(\cdot), i = 1, \ldots, N$, in (4.11) approximately assume the same value at $\boldsymbol{x}_\mu$, since they only depend on the distances $\left\|\boldsymbol{x}_\mu - \boldsymbol{x}_i\right\|_2$, namely:

$$w_1(\boldsymbol{x}_\mu) \approx \ldots \approx w_{N_1}(\boldsymbol{x}_\mu) \approx w_{N_1+1}(\boldsymbol{x}_\mu) \approx \ldots \approx w_N(\boldsymbol{x}_\mu) .$$

Finally, let us evaluate the gradient of the IDW distance function $\nabla_{\boldsymbol{x}} z_N(\cdot)$ in (5.1) at $\boldsymbol{x}_\mu$:

$$\begin{aligned}
\nabla_{\boldsymbol{x}} z_N(\boldsymbol{x}_\mu) &= -\frac{4}{\pi} \cdot \frac{\sum_{i=1}^N (\boldsymbol{x}_\mu - \boldsymbol{x}_i) \cdot w_i(\boldsymbol{x}_\mu)^2}{1 + \left[\sum_{i=1}^N w_i(\boldsymbol{x}_\mu)\right]^2} \\
&\qquad \text{As we have just seen, } w_i(\boldsymbol{x}_\mu) \approx w_1(\boldsymbol{x}_\mu), \forall i = 1, \ldots, N \\
&\approx -\frac{4}{\pi} \cdot \frac{\sum_{\boldsymbol{x}_i \in \mathcal{X}^{(1)}} \overbrace{(\boldsymbol{x}_\mu - \boldsymbol{x}_i)}^{\boldsymbol{x}_1 \approx \ldots \approx \boldsymbol{x}_{N_1}} \cdot w_1(\boldsymbol{x}_\mu)^2 + \sum_{\boldsymbol{x}_i \in \mathcal{X}^{(2)}} \overbrace{(\boldsymbol{x}_\mu - \boldsymbol{x}_i)}^{\boldsymbol{x}_{N_1+1} \approx \ldots \approx \boldsymbol{x}_N} \cdot w_1(\boldsymbol{x}_\mu)^2}{1 + \left[N \cdot w_1(\boldsymbol{x}_\mu)\right]^2} \\
&\approx -\frac{4}{\pi} \cdot \frac{w_1(\boldsymbol{x}_\mu)^2 \cdot \left[N_1 \cdot (\boldsymbol{x}_\mu - \boldsymbol{x}_1) + (N - N_1) \cdot (\boldsymbol{x}_\mu - \boldsymbol{x}_N)\right]}{1 + \left[N \cdot w_1(\boldsymbol{x}_\mu)\right]^2} \\
&\approx -\frac{4}{\pi} \cdot \frac{w_1(\boldsymbol{x}_\mu)^2 \cdot \left[N \cdot \boldsymbol{x}_\mu - N_1 \cdot \boldsymbol{x}_1 - (N - N_1) \cdot \boldsymbol{x}_N\right]}{1 + \left[N \cdot w_1(\boldsymbol{x}_\mu)\right]^2} \\
&\approx -\frac{4}{\pi} \cdot \frac{w_1(\boldsymbol{x}_\mu)^2 \cdot \left[N \cdot \frac{\boldsymbol{x}_1 + \boldsymbol{x}_N}{2} - N_1 \cdot \boldsymbol{x}_1 - (N - N_1) \cdot \boldsymbol{x}_N\right]}{1 + \left[N \cdot w_1(\boldsymbol{x}_\mu)\right]^2} \\
&\approx -\frac{4}{\pi} \cdot \frac{w_1(\boldsymbol{x}_\mu)^2}{1 + \left[N \cdot w_1(\boldsymbol{x}_\mu)\right]^2} \cdot \left[\left(\frac{N}{2} - N_1\right) \cdot \boldsymbol{x}_1 + \left(-\frac{N}{2} + N_1\right) \cdot \boldsymbol{x}_N\right] .
\end{aligned}$$

Clearly, if the clusters are nearly equally sized, i.e.

$$N_1 \approx N - N_1 \approx \frac{N}{2}, \tag{5.8}$$

then:

$$\nabla_{\boldsymbol{x}} z_N\left(\boldsymbol{x}_{\mu}\right) \approx \boldsymbol{0}_n.$$

Therefore, in this case, a stationary point for the IDW distance function $z_N(\boldsymbol{x})$ in (4.17) is approximately located at the midpoint between the centroids of the two clusters. The quality of this approximation depends on whether the assumptions in (5.7) and (5.8) are satisfied or not (and to what degree).

General case ($N > 2$). Any set of samples $\mathcal{X}$ in (2.9) can be partitioned into an arbitrary number of disjoint clusters, say $K \in \mathbb{N}, K \leq |\mathcal{X}| = N$, i.e.:

$$\mathcal{X} = \mathcal{X}^{(1)} \cup \mathcal{X}^{(2)} \cup \ldots \cup \mathcal{X}^{(K)}, \quad \text{such that } \mathcal{X}^{(i)} \cap \mathcal{X}^{(j)} = \emptyset, \forall i \neq j.$$

In this case, finding the local solutions of Problem (5.5) explicitly, or even approximately, is quite complex. Heuristically speaking, if the clusters are "well spread" (i.e. all the points contained inside each cluster $\mathcal{X}^{(i)}$ are sufficiently far away from the other ones in $\mathcal{X}^{(j)}$, $j = 1, \ldots, K$, $j \neq i$), then we can approximately deduce where the local minimizers of $z_N(\boldsymbol{x})$ in (4.17) are located. For instance, Figure 21 depicts a set of samples $\mathcal{X}$ that has been partitioned into three clusters, $\mathcal{X}^{(1)}, \mathcal{X}^{(2)}$ and $\mathcal{X}^{(3)}$, and for which the former hypothesis is satisfied. Let us write $z_N(\boldsymbol{x})$ in (4.17) by considering the contribution

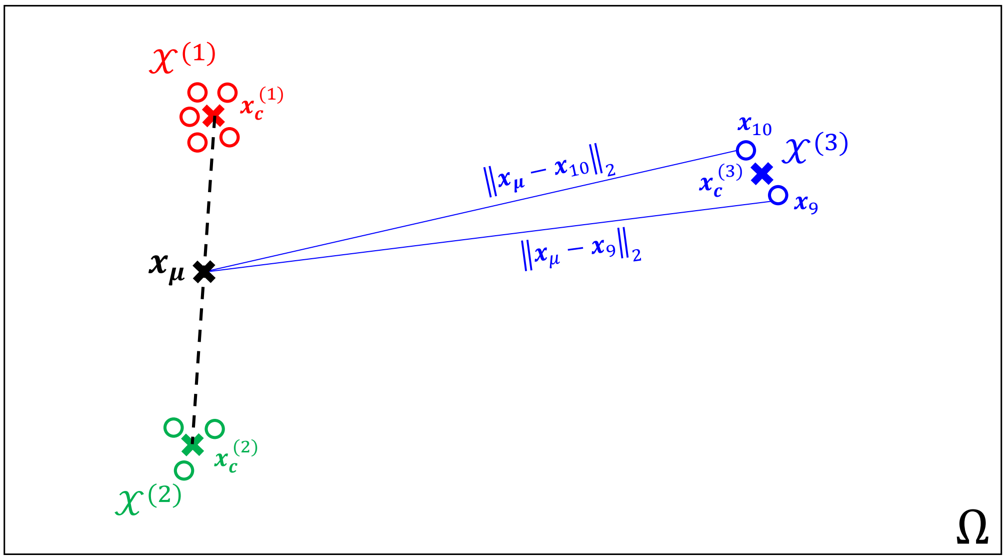


Figure 21: Two-dimensional example of "well spread" clusters, highlighted with different colors. The circles denote the points contained in $\mathcal{X}$ while the crosses represent the centroids of $\mathcal{X}^{(1)}, \mathcal{X}^{(2)}$ and $\mathcal{X}^{(3)}$. $\boldsymbol{x}_\mu$ is the midpoint between the centroids of clusters $\mathcal{X}^{(1)}$ and $\mathcal{X}^{(2)}$. Finally, the blue lines highlight the distances between the samples of cluster $\mathcal{X}^{(3)}$ and $\boldsymbol{x}_\mu$.

of each cluster explicitly and in the case $\boldsymbol{x} \in \mathbb{R}^n \setminus \mathcal{X}$:

$$z_N(\boldsymbol{x}) = -\frac{2}{\pi} \cdot \arctan\left[\left(\sum_{\boldsymbol{x}_i \in \mathcal{X}^{(1)}} \frac{1}{\|\boldsymbol{x} - \boldsymbol{x}_i\|_2^2} + \sum_{\boldsymbol{x}_i \in \mathcal{X}^{(2)}} \frac{1}{\|\boldsymbol{x} - \boldsymbol{x}_i\|_2^2} + \sum_{\boldsymbol{x}_i \in \mathcal{X}^{(3)}} \frac{1}{\|\boldsymbol{x} - \boldsymbol{x}_i\|_2^2}\right)^{-1}\right].$$

In the general case, given the clusters $\mathcal{X}^{(i)}$ and $\mathcal{X}^{(j)}$, with a slight abuse of notation, we define their centroids $\boldsymbol{x}_{\boldsymbol{c}}^{(i)}, \boldsymbol{x}_{\boldsymbol{c}}^{(j)}$, and the corresponding midpoint $\boldsymbol{x}_\mu$ between them as:

$$\boldsymbol{x}_{\boldsymbol{c}}^{(k)} = \frac{\sum_{\boldsymbol{x}_i \in \mathcal{X}^{(k)}} \boldsymbol{x}_i}{\left|\mathcal{X}^{(k)}\right|} \qquad \text{centroid of the } k\text{-th cluster,} \tag{5.9a}$$

$$\boldsymbol{x}_\mu = \frac{\boldsymbol{x}_{\boldsymbol{c}}^{(i)} + \boldsymbol{x}_{\boldsymbol{c}}^{(j)}}{2} \qquad \text{midpoint.} \tag{5.9b}$$

Going back to the example depicted in Figure 21, if we consider the midpoint $\boldsymbol{x}_\mu$ between the centroids of $\mathcal{X}^{(1)}$ and $\mathcal{X}^{(2)}$, due to the "well spread" hypothesis we can say that $\left\|\boldsymbol{x}_\mu - \boldsymbol{x}_i\right\|_2 \gg 0, \forall \boldsymbol{x}_i \in \mathcal{X}^{(3)}$, making the contributions of the points inside the third cluster negligible when evaluating $z_N(\boldsymbol{x})$ in (4.17) at $\boldsymbol{x}_\mu$. Formally:

$$z_N\left(\boldsymbol{x}_\mu\right) \approx -\frac{2}{\pi} \cdot \arctan\left[\left(\sum_{\boldsymbol{x}_i \in \mathcal{X}^{(1)}} \frac{1}{\left\|\boldsymbol{x}_\mu - \boldsymbol{x}_i\right\|_2^2} + \sum_{\boldsymbol{x}_i \in \mathcal{X}^{(2)}} \frac{1}{\left\|\boldsymbol{x}_\mu - \boldsymbol{x}_i\right\|_2^2}\right)^{-1}\right].$$

Similarly, its gradient at $\boldsymbol{x}_\mu$ is approximately equal to:

$$\nabla_{\boldsymbol{x}} z_N\left(\boldsymbol{x}_\mu\right) \approx -\frac{4}{\pi} \cdot \frac{\sum_{\boldsymbol{x}_i \in \mathcal{X}^{(1)}} \frac{\boldsymbol{x}_\mu - \boldsymbol{x}_i}{\left\|\boldsymbol{x}_\mu - \boldsymbol{x}_i\right\|_2^4} + \sum_{\boldsymbol{x}_i \in \mathcal{X}^{(2)}} \frac{\boldsymbol{x}_\mu - \boldsymbol{x}_i}{\left\|\boldsymbol{x}_\mu - \boldsymbol{x}_i\right\|_2^4}}{1 + \left[\sum_{\boldsymbol{x}_i \in \mathcal{X}^{(1)}} \frac{1}{\left\|\boldsymbol{x}_\mu - \boldsymbol{x}_i\right\|_2^2} + \sum_{\boldsymbol{x}_i \in \mathcal{X}^{(2)}} \frac{1}{\left\|\boldsymbol{x}_\mu - \boldsymbol{x}_i\right\|_2^2}\right]^2}.$$

In general, given K clusters, if these are "well spread", then we can consider each possible couple of clusters separately and neglect the contributions of the remaining ones. Approximately speaking, we could split the general case into $\binom{K}{2}$ distinct problems that read as follows: find the stationary points of the IDW distance function $z_{N_{i\cup j}}(\boldsymbol{x})$ in (4.17) defined from the set of samples $\mathcal{X}^{(i)} \cup \mathcal{X}^{(j)}, i \neq j$ and $N_{i\cup j} = \left|\mathcal{X}^{(i)} \cup \mathcal{X}^{(j)}\right|$. Hence, rough locations of the stationary points of $z_{N_{i\cup j}}(\boldsymbol{x})$ can be found by following the same rationale proposed for the previously analyzed cases.

The following Remark summarizes the results obtained in this Section.

Remark 5.2. *Consider the IDW distance function $z_N(\boldsymbol{x})$ in* (4.17) *defined from a set of samples $\mathcal{X}$ in* (2.9)*, $|\mathcal{X}| = N$. We have that:*

1. *The stationary points of $z_N(\boldsymbol{x})$ are approximately located at the midpoints between the centroids of different clusters of $\mathcal{X}$. In case $\mathcal{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2\}, N = 2$, then the midpoint $\boldsymbol{x}_\mu = \frac{\boldsymbol{x}_1 + \boldsymbol{x}_2}{2}$ is an exact stationary point for the IDW distance function. Vice-versa, if $N > 2$, the quality of this approximation depends on: (i) the distance between the points inside each cluster, (ii) the number of samples which constitute each subset of $\mathcal{X}$ and (iii) how "well" the clusters are spread on Ω (i.e. the points inside each cluster must be sufficiently far away from the samples contained in the other subsets of $\mathcal{X}$).*

2. *The solutions of Problem* (5.5) *might be located at the vertices of the box defined by the constraints* $\{\boldsymbol{x} : \boldsymbol{l} \leq \boldsymbol{x} \leq \boldsymbol{u}\}$.

Some scalar examples of all the previously analyzed situations are reported in Figure 20. Note that, although the approximate locations of the stationary points of $z_N(\boldsymbol{x})$ in (4.17) described in Remark 5.2 seem to depend on many simplifying assumptions, this information is sufficient to make $\hat{f}_N(\boldsymbol{x})$ in (4.1) and the IDW distance function in (4.17) assume comparable ranges (after min-max rescaling), as we will see in the next Section.

5.2 A revisited infill sampling criterion

In this Section, we take advantage of the results summarized in Remark 5.2 to revisit the infill sampling criteria of `GLIS` [10] and `GLISp` [11]. In particular, we build an augmented sample set $\mathcal{X}_{aug} \supset \mathcal{X}$ based on the approximate locations of the solutions of Problem (5.5) and use it to rescale the surrogate model $\hat{f}_N(\boldsymbol{x})$ in (4.1) and the IDW distance function $z_N(\boldsymbol{x})$ in (4.17). We propose an acquisition function that is a weighted sum between these two contributions. Furthermore, differently from `GLIS` [10] and `GLISp` [11], we suggest to cycle the exploration-exploitation trade-off weight. The same infill sampling criteria is used both for BBO and PBO.

5.2.1 Min-max rescaling and augmented sample set

We start by defining how min-max rescaling operates.

Definition 5.1: Min-max rescaling [56]. *Given a set of samples:*

$$\mathcal{X} = \{\boldsymbol{x}_i : i = 1, \dots, N, \boldsymbol{x}_i \in \mathbb{R}^n\}$$

and a generic multivariable function $h : \mathbb{R}^n \to \mathbb{R}$*, min-max rescaling (or min-max normalization) rescales* $h(\boldsymbol{x})$ *to* $\bar{h} : \mathbb{R}^n \to \mathbb{R}$*, defined as:*

$$\bar{h}(\boldsymbol{x}; \mathcal{X}) = \frac{h(\boldsymbol{x}) - h_{min}(\mathcal{X})}{\Delta H(\mathcal{X})}, \tag{5.10}$$

where[a]*:*

$$h_{min}(\mathcal{X}) = \min_{\boldsymbol{x}_i \in \mathcal{X}} h(\boldsymbol{x}_i), \tag{5.11a}$$

$$h_{max}(\mathcal{X}) = \max_{\boldsymbol{x}_i \in \mathcal{X}} h(\boldsymbol{x}_i), \tag{5.11b}$$

$$\Delta H(\mathcal{X}) = h_{max}(\mathcal{X}) - h_{min}(\mathcal{X}). \tag{5.11c}$$

[a]Note that, to avoid dividing by zero in (5.10), $\Delta H(\mathcal{X})$ can be set to $h_{max}(\mathcal{X})$ or 1 whenever $h_{min}(\mathcal{X}) = h_{max}(\mathcal{X}) \neq 0$ or $h_{min}(\mathcal{X}) = h_{max}(\mathcal{X}) = 0$ respectively.

The objective of min-max rescaling is to obtain a function with range $[0, 1]$, i.e. we would like to have $\bar{h} : \mathbb{R}^n \to [0, 1]$. Clearly, the quality of the normalization depends on the information brought by the samples contained inside $\mathcal{X}$, as pointed out in the following Remark.

Remark 5.3 (Quality of min-max rescaling)**.** *We can observe that:*

1. *If $\mathcal{X}$ contains the global minimizer(s) and maximizer(s) of $h(\boldsymbol{x})$, then $\bar{h}(\boldsymbol{x})$ defined as in* (5.10) *effectively has range* $[0, 1]$*,*
2. *Otherwise, we can only ensure that* $0 \leq \bar{h}(\boldsymbol{x}_i) \leq 1, \forall \boldsymbol{x}_i \in \mathcal{X}$.
3. *In general, if we increase the number of distinct samples in $\mathcal{X}$, then the rescaling of $h(\boldsymbol{x})$ gets better (or, worst case, stays the same).*

Going back to the problem of rescaling the IDW distance function $z_N(\boldsymbol{x})$ in (4.17), if we were to apply (5.10) using the set of previously evaluated samples $\mathcal{X}$ in (2.9), then it would not be effective since $z_N(\boldsymbol{x}_i) = 0, \forall \boldsymbol{x}_i \in \mathcal{X}$ (see Proposition 5.1). Instead, we have opted to generate a sufficiently expressive augmented sample set $\mathcal{X}_{aug} \supset \mathcal{X}$ and perform min-max normalization using $\mathcal{X}_{aug}$ instead of $\mathcal{X}$.
Consider the general case described in Section 5.1.2. Then, the augmented sample set $\mathcal{X}_{aug}$ can be built in the following fashion (see Remark 5.2):

1. Partition the points in $\mathcal{X}$ (2.9) into different clusters. Here, for simplicity, we fix a-priori the number $K_{aug} \in \mathbb{N}$ of clusters and apply *K-means clustering* [15, 58] to obtain the sets $\mathcal{X}^{(1)}, \ldots, \mathcal{X}^{(K_{aug})}$;
2. Compute the centroids of each cluster, using (5.9a), and group them inside the set $\mathcal{X}_c = \left\{\boldsymbol{x_c}^{(1)}, \ldots, \boldsymbol{x_c}^{(K_{aug})}\right\}$;
3. Calculate all the midpoints $\boldsymbol{x_\mu}$ between each possible couple of centroids $\boldsymbol{x_c}^{(i)}, \boldsymbol{x_c}^{(j)} \in \mathcal{X}_c, \boldsymbol{x_c}^{(i)} \neq \boldsymbol{x_c}^{(j)}$, using (5.9b);
4. Build the augmented sample set as $\mathcal{X}_{aug} = \mathcal{X} \cup \mathcal{X}_\mu$, where $\mathcal{X}_\mu$ is the set which groups all the previously computed midpoints.

Clearly, as highlighted by (5.10) and Remark 5.3, if $\mathcal{X}_{aug}$ contains points that are close (or equal) to the global minimizers and maximizers of $z_N(\boldsymbol{x})$ in (4.17), then the quality of the min-max rescaling of the IDW distance function improves.
Algorithm 13 formalizes the aforementioned steps while also taking into consideration the case $|\mathcal{X}| \leq K_{aug}$ (for which no clustering is performed). Note that we also include the bounds $\boldsymbol{l}$ and $\boldsymbol{u}$

inside $\mathcal{X}_c$ and $\mathcal{X}_{aug}$ for two reasons: (i) $\boldsymbol{l}$ or $\boldsymbol{u}$ might actually be the solutions of Problem (5.5) (see Remark 5.2)[1] and (ii) given that we also want to rescale $\hat{f}_N(\boldsymbol{x})$ in (4.1), adding additional points to the augmented sample set improves the overall quality of the min-max normalization (see Remark 5.3). Notice that the number of points contained inside $\mathcal{X}_{aug}$ obtained from Algorithm 13 is:

$$\left|\mathcal{X}_{aug}\right| = |\mathcal{X}| + \binom{K_{aug}+2}{2} + 2.$$

Therefore, to avoid excessively large augmented sample sets, K_{aug} needs to be chosen appropriately. Empirically, after many tests, we have found out that $K_{aug} = 5$ gets the job done for most optimization problems (see Chapter 7).

Algorithm 13: Computation of $\mathcal{X}_{aug}$ for min-max rescaling

Input: (i) Set of samples $\mathcal{X}$ in (2.9); (ii) Number of clusters $K_{aug} \in \mathbb{N}$; (iii) Lower bounds $\boldsymbol{l} \in \mathbb{R}^n$ and upper bounds $\boldsymbol{u} \in \mathbb{R}^n$ of the GOP (2.1).

Output: (i) Augmented sample set $\mathcal{X}_{aug} \supset \mathcal{X}$.

```
1: if $|\mathcal{X}| > K_{aug}$ then
2:     Perform $K$-means clustering [15, 58] to group the samples in $\mathcal{X}$ into $K_{aug}$ clusters $\mathcal{X}^{(1)}, \dots, \mathcal{X}^{(K_{aug})}$
3:     Compute the set of centroids $\mathcal{X}_c$ using (5.9a):
       $$\mathcal{X}_c = \left\{\boldsymbol{x}_{\boldsymbol{c}}^{(k)} : \boldsymbol{x}_{\boldsymbol{c}}^{(k)} = \frac{\sum_{\boldsymbol{x}_i \in \mathcal{X}^{(k)}} \boldsymbol{x}_i}{\left|\mathcal{X}^{(k)}\right|}, k = 1, \dots, K_{aug}\right\}$$
4: else
5:     Set $\mathcal{X}_c = \mathcal{X}$
6: Add the bounds to $\mathcal{X}_c$: $\mathcal{X}_c = \mathcal{X}_c \cup \{\boldsymbol{l}, \boldsymbol{u}\}$
7: Group all possible couples of $\mathcal{X}_c$ (without repetition):
   $$\mathcal{X}_{couples} = \left\{\left(\boldsymbol{x}_{\boldsymbol{c}}^{(i)}, \boldsymbol{x}_{\boldsymbol{c}}^{(j)}\right) : \boldsymbol{x}_{\boldsymbol{c}}^{(i)}, \boldsymbol{x}_{\boldsymbol{c}}^{(j)} \in \mathcal{X}_c, \boldsymbol{x}_{\boldsymbol{c}}^{(i)} \neq \boldsymbol{x}_{\boldsymbol{c}}^{(j)}\right\}$$
8: Calculate the midpoints between all the couples inside $\mathcal{X}_{couples}$, obtaining the set:
   $$\mathcal{X}_{\mu} = \left\{\boldsymbol{x}_{\mu} : \boldsymbol{x}_{\mu} = \frac{\boldsymbol{x}_{\boldsymbol{c}}^{(i)} + \boldsymbol{x}_{\boldsymbol{c}}^{(j)}}{2}, \left(\boldsymbol{x}_{\boldsymbol{c}}^{(i)}, \boldsymbol{x}_{\boldsymbol{c}}^{(j)}\right) \in \mathcal{X}_{couples}\right\}$$
9: Build the augmented sample set as $\mathcal{X}_{aug} = \mathcal{X} \cup \mathcal{X}_{\mu} \cup \{\boldsymbol{l}, \boldsymbol{u}\}$
```

As a final remark, we point out that we could perform min-max normalization in (5.10) by using the real minima and maxima of $z_N(\boldsymbol{x})$ in (4.17) and $\hat{f}_N(\boldsymbol{x})$ in (4.1), which can be obtained by solving four additional global optimization problems. However, we have preferred to stick with the proposed heuristic way since we are not interested in an extremely accurate rescaling and, also, to avoid potentially high overhead times due to solving additional optimization problems.

[1]We could add all 2^n vertices of the box defined by the bound constraints $\{\boldsymbol{x} : \boldsymbol{l} \leq \boldsymbol{x} \leq \boldsymbol{u}\}$. However, we have preferred to include only $\boldsymbol{l}$ and $\boldsymbol{u}$ to avoid increasing the cardinality of the augmented sample set, especially in the case of high-dimensional problems.

5.2.2 Definition of the acquisition function

We propose to use the same acquisition function both for black-box and preference-based optimization. In particular, we define $a_N(\boldsymbol{x})$ as a weighted sum between the surrogate model $\hat{f}_N(\boldsymbol{x})$ in (4.1) and the IDW distance function $z_N(\boldsymbol{x})$ in (4.17). The two contributions are rescaled using min-max normalization (see Definition 5.1), based on the augmented sample set $\mathcal{X}_{aug}$ generated by Algorithm 13. We refer to the resulting surrogate-based methods as `GLIS-r` [108] and `GLISp-r` [109], since they use the surrogate models and the exploration function of `GLIS` [10] and `GLISp` [11] respectively, but adopt a different acquisition function based on min-max rescaling (hence the `-r`).

Definition 5.2: Acquisition function for `GLIS-r` [108] and `GLISp-r` [109]. *The aforementioned methods adopt an infill sampling criterion that is based on the following acquisition function $a_N : \mathbb{R}^n \to \mathbb{R}$:*

$$a_N(\boldsymbol{x}) = \delta \cdot \bar{\hat{f}}_N\left(\boldsymbol{x}; \mathcal{X}_{aug}\right) + (1-\delta) \cdot \bar{z}_N\left(\boldsymbol{x}; \mathcal{X}_{aug}\right), \tag{5.12}$$

where $\hat{f}_N(\boldsymbol{x})$ in (4.1) *and $z_N(\boldsymbol{x})$ in* (4.17) *have been rescaled using min-max normalization as in* (5.10) *and $\mathcal{X}_{aug}$ is generated by Algorithm 13. Finally, $\delta \in [0, 1]$ is the exploration-exploitation trade-off weight.*

Analogously to `GLIS` [10] and `GLISp` [11], `GLIS-r` [108] and `GLISp-r` [109] look for new candidate samples by minimizing the acquisition function $a_N(\boldsymbol{x})$ in (5.12), i.e.:

$$\begin{aligned} \boldsymbol{x}_{N+1} = \arg\min_{\boldsymbol{x}} &\; a_N(\boldsymbol{x}) \\ \text{s.t.} &\quad \boldsymbol{x} \in \Omega. \end{aligned} \tag{5.13}$$

Note that setting $\delta = 0$ for $a_N(\boldsymbol{x})$ in (5.12) corresponds to *pure exploration*, while fixing $\delta = 1$ results in *pure exploitation*.

The next Proposition and Lemma address the differentiability of $a_N(\boldsymbol{x})$ in (5.12).

***Proposition 5.2: Differentiability of the acquisition function in* (5.12).** *The acquisition function $a_N(\boldsymbol{x})$ in* (5.12) *is differentiable everywhere if and only if the chosen radial basis function $\phi_{f_i}(\boldsymbol{x}; \epsilon_f) = \varphi_f(\epsilon_f \cdot \|\boldsymbol{x} - \boldsymbol{x}_i\|_2)$ for the surrogate model $\hat{f}_N\left(\boldsymbol{x}; \boldsymbol{\beta}_f, \epsilon_f\right)$ in* (4.1) *is differentiable everywhere.*

Proof. The aforementioned result follows immediately from the application of Propositions 2.1 and 4.4, which cover the differentiability of each term that appears in the weighted sum in (5.12). □

Lemma 5.2: Gradient of the acquisition function in (5.12)**.** *Suppose that $a_N(\boldsymbol{x})$ in* (5.12) *is differentiable everywhere (see Proposition 5.2). Then, its gradient is:*

$$\nabla_{\boldsymbol{x}} a_N(\boldsymbol{x}) = \frac{\delta}{\Delta \hat{F}_N(\mathcal{X}_{aug})} \cdot \nabla_{\boldsymbol{x}} \hat{f}_N(\boldsymbol{x}) + \frac{1-\delta}{\Delta Z_N(\mathcal{X}_{aug})} \cdot \nabla_{\boldsymbol{x}} z_N(\boldsymbol{x}), \tag{5.14}$$

where $\nabla_{\boldsymbol{x}} \hat{f}_N(\boldsymbol{x})$ and $\nabla_{\boldsymbol{x}} z_N(\boldsymbol{x})$ are defined respectively as in (2.22) *and* (5.1).

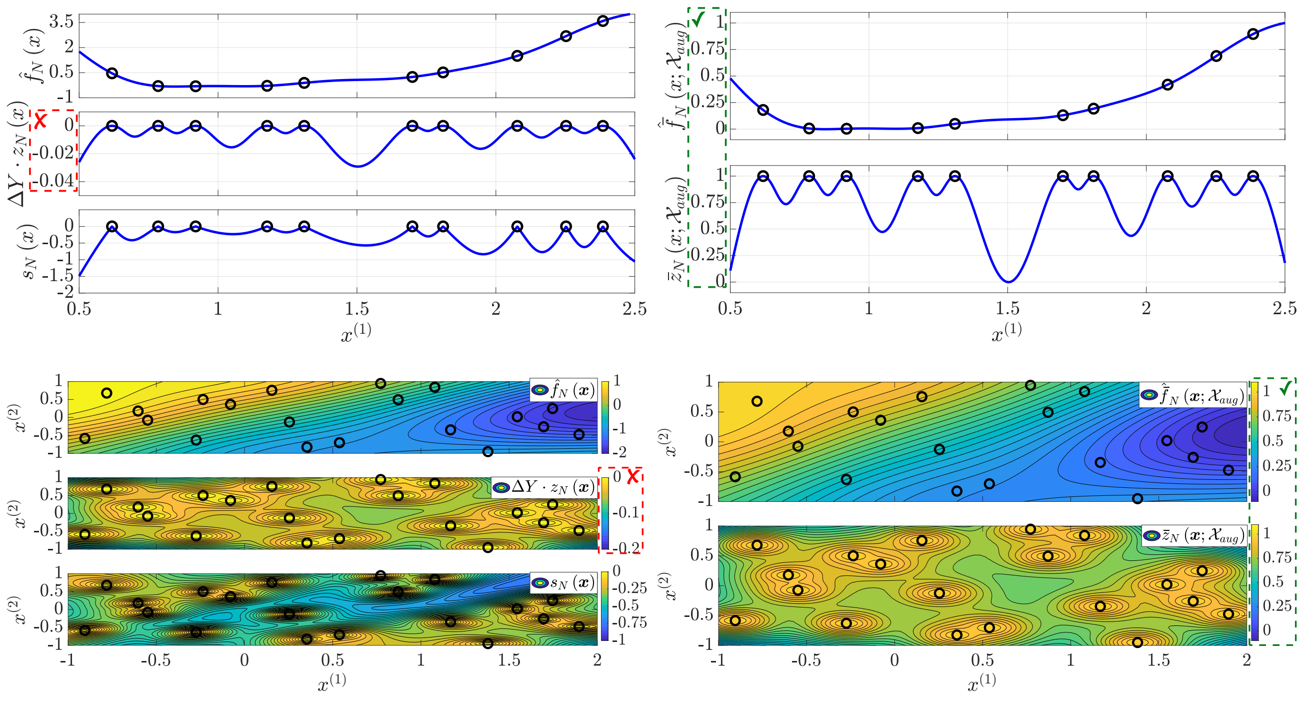
Proof. The Proof is straightforward. $a_N(\boldsymbol{x})$ in (5.12) is a weighted sum of differentiable functions, $\hat{f}_N(\boldsymbol{x})$ in (4.1) and $z_N(\boldsymbol{x})$ in (4.17), hence its gradient is simply the weighted sum of their gradients. □

The proposed acquisition function in Definition 5.2 is similar to $a_N(\boldsymbol{x})$ of `MSRS` [116] (cf. (2.52)) but here we use an ad-hoc augmented sample set $\mathcal{X}_{aug}$ (instead of a randomly generated one $\mathcal{X}_\Omega$) and a different exploration function. Furthermore, `MSRS` [116] is used only for black-box optimization, whereas we employ the acquisition function $a_N(\boldsymbol{x})$ in (5.12) both for BBO and PBO (we simply need to "change" the surrogate model).

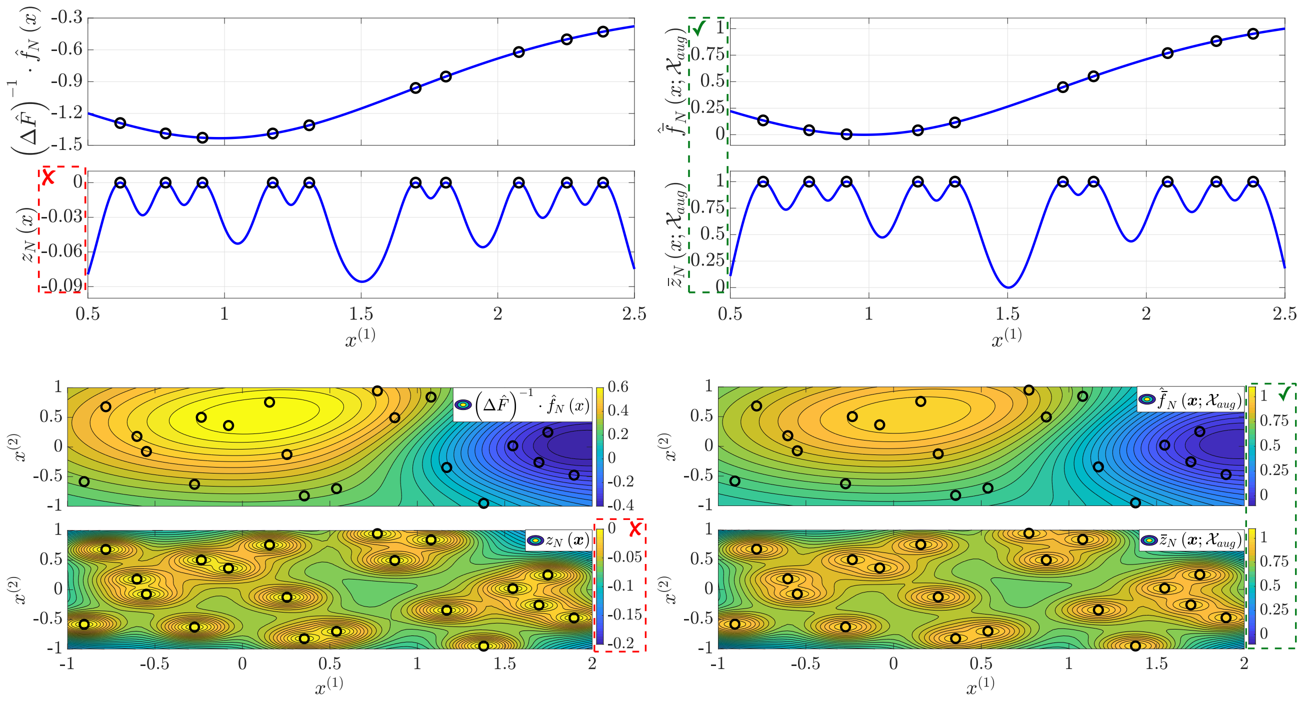
Table 1: Comparison of the acquisition functions of the `GLIS` [10], `GLISp` [11], `GLIS-r` [108] and `GLISp-r` [109] algorithms.

	`GLIS` [10] and `GLISp` [11]	`GLIS-r` [108] and `GLISp-r` [109]
BBO	$\hat{f}_N(\boldsymbol{x}) + \Delta Y \cdot \delta_1 \cdot z_N(\boldsymbol{x}) + \delta_2 \cdot s_N(\boldsymbol{x})$	$\delta \cdot \hat{\hat{f}}_N(\boldsymbol{x}; \mathcal{X}_{aug}) + (1-\delta) \cdot \bar{z}_N(\boldsymbol{x}; \mathcal{X}_{aug})$
PBO	$(\Delta\hat{F})^{-1} \cdot \hat{f}_N(\boldsymbol{x}) + \delta \cdot z_N(\boldsymbol{x})$	
	$\Delta Y = \max\{\max_{y_i \in \mathcal{Y}} y_i - \min_{y_i \in \mathcal{Y}} y_i, \epsilon_{\Delta Y}\}$	$\mathcal{X}_{aug}$ generated by Algorithm 13
	$\Delta\hat{F} = \max_{\boldsymbol{x}_i \in \mathcal{X}} \hat{f}_N(\boldsymbol{x}_i) - \min_{\boldsymbol{x}_i \in \mathcal{X}} \hat{f}_N(\boldsymbol{x}_i)$	$\hat{\hat{f}}_N(\cdot)$ and $\bar{z}_N(\cdot)$ rescaled as in (5.10)
	$\delta_1, \delta_2, \delta \in \mathbb{R}_{\geq 0}, \epsilon_{\Delta Y} \in \mathbb{R}_{>0}$	$\delta \in [0, 1]$

Table 1 compares the acquisition functions of `GLIS` [10], `GLISp` [11], `GLIS-r` [108] and `GLISp-r` [109]. A visual comparison of each term of $a_N(\boldsymbol{x})$ in (4.20a), (4.20b) and (5.12) is presented in Figure 22. Notice how, *even though either the surrogate model in* (4.1) *or the IDW distance function in* (4.17) *are rescaled by ΔY (`GLIS` [10]) or $\Delta\hat{F}$ (`GLISp` [11]), their ranges differ by one or even two orders of magnitudes*. Hence, unless δ_1 in (4.20a) or δ in (4.20b) are properly chosen, $z_N(\boldsymbol{x})$ and $\hat{f}_N(\boldsymbol{x})$ are not directly comparable. Furthermore, as we have proven in Section 5.1.1, the values assumed by the IDW distance function in (4.17) tend to decrease as the number of samples, N, increases. Therefore, δ_1 and δ of `GLIS` [10] and `GLISp` [11] should be iteratively varied to compensate for this shortcoming. Recall that the convergence of any global optimization algorithm is strictly related to how the exploration is performed (see Chapters 1, 2 and 3). As we will see in Chapter 7, *compared to `GLISp-r` [109], algorithm `GLISp` [11] is more prone to getting stuck on the local minima of several benchmark GOPs* (2.1), due to the shortcomings of $z_N(\boldsymbol{x})$ in (4.17), discussed in Section 5.1.1. Instead, `GLIS` [10] is less affected by the limitations of the IDW distance function in (4.17), since most of the exploration is carried out by $s_N(\boldsymbol{x})$ in (4.19).



(a) `GLIS` [10] vs `GLIS-r` [108] (black-box optimization).



(b) `GLISp` [11] vs `GLISp-r` [109] (preference-based optimization).

Figure 22: Comparison between the terms in the acquisition functions reported in Table 1 (left: original acquisition functions, right: proposed one). The one-dimensional examples are obtained from the `gramacy and lee` [53] function and $N = 10$, while the two-dimensional examples result from the `adjiman` [62] function and $N = 20$ (see Appendix B). For `GLIS-r` [108] and `GLISp-r` [109], the number of centroids used to build $\mathcal{X}_{aug}$ through Algorithm 13 is $K_{aug} = 5$.

We conclude this Section by pointing out that the selection of $\delta \in [0, 1]$ for the proposed acquisition function $a_N(\boldsymbol{x})$ in (5.12) is more intuitive compared to the choice of the exploration-exploitation trade-off weights for `GLIS` [10] and `GLISp` [11] (see Table 1). That is due to the fact that $\hat{\bar{f}}_N\left(\boldsymbol{x}; \mathcal{X}_{aug}\right)$ and $\bar{z}_N\left(\boldsymbol{x}; \mathcal{X}_{aug}\right)$ in (5.12) roughly share the same range, which is $[0, 1]$. In the following Section, we address the selection of δ much more in detail.

5.2.3 Greedy δ-cycling

As highlighted in Section 2.6, many black-box optimization methods iteratively vary the exploration-exploitation trade-off weights for their respective infill sampling criteria. In the original formulations of `GLIS` [10] and `GLISp` [11], hyper-parameters δ_1 and δ_2 for $a_N(\boldsymbol{x})$ in (4.20a), as well as δ for $a_N(\boldsymbol{x})$ in (4.20b), are kept constant throughout the whole optimization process. Also, defining some form of cycling for such coefficients can be quite complex given that, as previously seen in Section 5.2.2, the additive terms in $a_N(\boldsymbol{x})$ are not always comparable. In this book, we suggest to cycle δ for the acquisition function in (5.12) following a strategy that is in between that of `MSRS` [116] and `SO-SA` [151]. We refer to the proposed strategy as greedy δ-cycling, which operates as follows. We define a set of $N_{cycle} \in \mathbb{N}$ weights to cycle (cycling set):

$$\Delta_{cycle} = \langle \delta_0, \dots, \delta_{N_{cycle}-1} \rangle, \quad \delta_j \in [0, 1], \forall \delta_j \in \Delta_{cycle}. \tag{5.15}$$

The set Δ_{cycle} should contain values that are well spread within the $[0, 1]$ range as to properly alternate between local and global search. As long as the best candidate sample $\boldsymbol{x_{best}}(N)$ found by `GLIS-r` [108] or `GLISp-r` [109] improves from an iteration to the other (see Algorithm 9), hyper-parameter δ in (5.12) is kept unchanged. Vice-versa, whenever `GLIS-r` [108] or `GLISp-r` [109] produce a new candidate sample $\boldsymbol{x}_{N+1}$ (by solving Problem (5.13)) that does not improve upon $\boldsymbol{x_{best}}(N)$, the weight is cycled following the order proposed in Δ_{cycle}. More formally, suppose that, at iteration k, we have at our disposal $|\mathcal{X}| = N$ samples and denote the trade-off parameter δ in (5.12) as $\delta(k)$ to highlight the iteration number. Furthermore, assume $\delta(k) = \delta_j \in \Delta_{cycle}$, which has been used to find the new candidate sample $\boldsymbol{x}_{N+1}$ at iteration k by solving Problem (5.13). Then, at iteration $k+1$, we select $\delta(k+1) \in \Delta_{cycle}$ as:

$$\delta(k+1) = \begin{cases} \delta(k) & \text{if } y_{N+1} < y_{best}(N) \text{ (BBO) or } \boldsymbol{x}_{N+1} \succ \boldsymbol{x_{best}}(N) \text{ (PBO)} \\ \delta_{(j+1) \bmod N_{cycle}} & \text{otherwise} \end{cases}. \tag{5.16}$$

Note that, in unconstrained BBO and PBO, a new candidate sample $\boldsymbol{x}_{N+1}$ improves upon $\boldsymbol{x}_{best}(N)$ either if it achieves a lower cost or if it is preferred to the latter (cf. Algorithm 9). Instead, in constrained BBO and PBO, we should also consider the Ξ-feasibility of $\boldsymbol{x}_{best}(N)$ and $\boldsymbol{x}_{N+1}$ (as we will see in Section 6.2.1).

The convergence of GLIS-r [108] and GLISp-r [109] is strictly related to the choice of the cycling set Δ_{cycle} in (5.15) and will be covered in the next Section.

5.3 Algorithms and convergence

Algorithm 14 formalizes the GLIS-r [108] and GLISp-r [109] procedures. We use the same Algorithm to describe black-box and preference-based optimization jointly. The only difference between the two is the information available on $f(\boldsymbol{x})$ of the GOP (2.1) and, consequently, how the surrogate models are estimated. GLIS-r [108] and GLISp-r [109] use the same infill sampling criteria, based on the acquisition function $a_N(\boldsymbol{x})$ in (5.12) and greedy δ-cycling (described in Section 5.2.3). For the sake of clarity, in Algorithm 14 we highlight in grey how GLIS-r [108] and GLISp-r [109] differ from GLIS [10] (Algorithm 10) and GLISp [11] (Algorithm 11).
The next Example highlights the advantages of the greedy δ-cycling strategy (in Section 5.2.3) over the non-greedy alternative (used by most BBO methods, cf. Section 2.6).

Example 5.1: Advantages of the greedy δ-cycling strategy

Consider the `adjiman` [62] benchmark optimization problem in Appendix B. We perform black-box optimization using Algorithm 14 (GLIS-r [108]) with hyper-parameters:

- Number of initial samples $N_{init} = 4$,
- Budget $N_{max} = 20$,
- Hyper-parameters for the surrogate model $\hat{f}_N(\boldsymbol{x})$ in (4.1): $\epsilon_f = 0.5378$, $\varphi_f(\cdot)$ inverse quadratic, $\epsilon_{SVD} = 10^{-6}$,
- Cycling set $\Delta_{cycle} = \langle 0.95, 0.5, 0 \rangle$,
- Number of clusters for $\mathcal{X}_{aug}$: $K_{aug} = 5$,
- No recalibration of $\hat{f}_N(\boldsymbol{x})$ in (4.1), i.e. $\mathcal{K}_{R_f} = \emptyset$, $\mathcal{M}_f = \emptyset$.

Problem (5.13) is solved by the PSWARM [72] algorithm (see Section 1.2.5). We compare the performances of the greedy δ-cycling strategy (Section 5.2.3) with the more traditional approach of alternating the exploration-exploitation trade-off weight at each iteration, regardless of the improvement. Figure 23 depicts the results obtained by Algorithm 14 with and without greedy δ-cycling. Notice how *the greedy approach converges to the global minimizer of the GOP* (2.1) *faster than the non-greedy alternative, since it keeps exploiting the surrogate model ($\delta = 0.95$) as long as there is an improvement*.

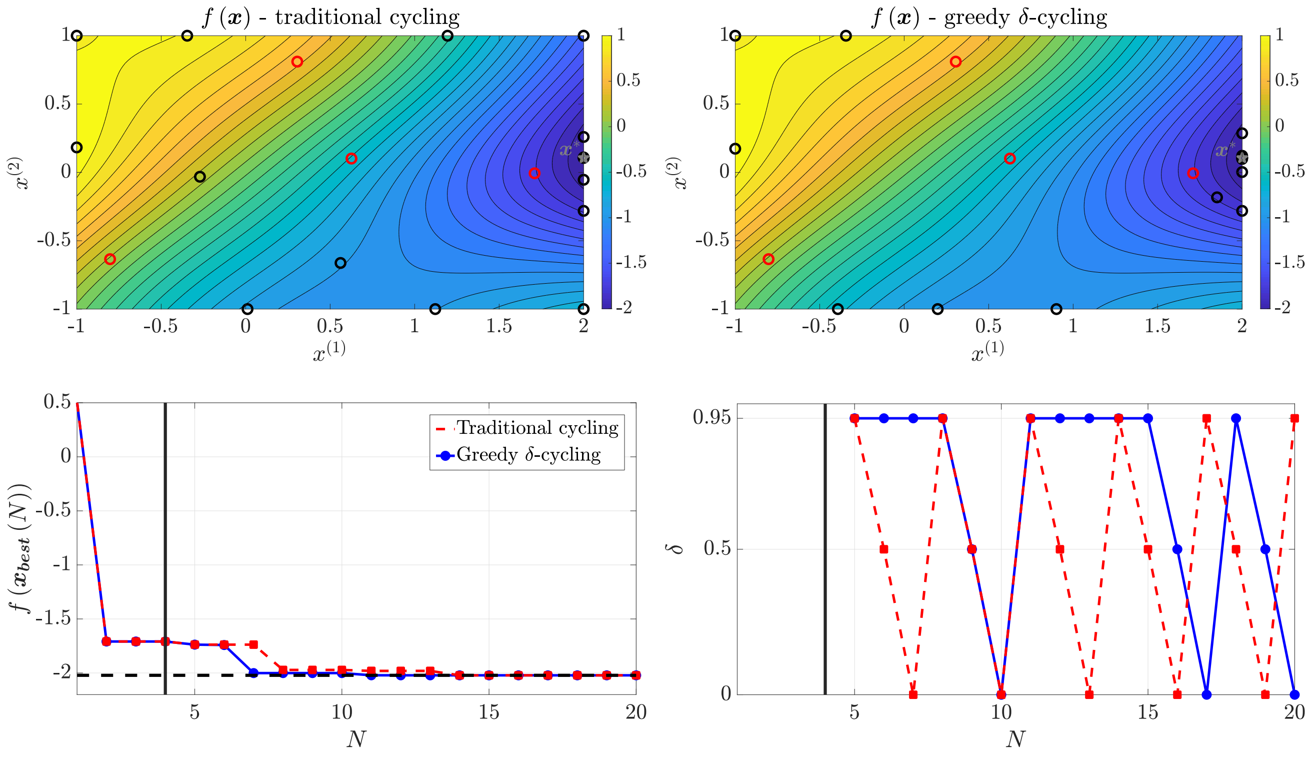


Figure 23: Comparison of the performances of `GLIS-r` [108] equipped with the greedy δ-cycling strategy versus the non-greedy approach, as described in Example 5.1. The top row displays the samples obtained by the BBO procedures (circles). The red circles are the N_{init} initial samples, whereas the ones in black are obtained by solving Problem (5.13). The star in grey is the global minimizer of the GOP (2.1). On the bottom left: performances achieved by `GLIS-r` [108] with and without greedy δ-cycling. The dashed black-line is the global minima of the GOP (2.1). The black vertical line denotes N_{init}. On the bottom right: parameter δ used to find each $\boldsymbol{x}_N, N > N_{init}$, when solving Problem (5.13).

Next, we compare the performances of `GLIS-r` [108] and `GLISp-r` [109] for different choices of the cycling set Δ_{cycle} in (5.15).

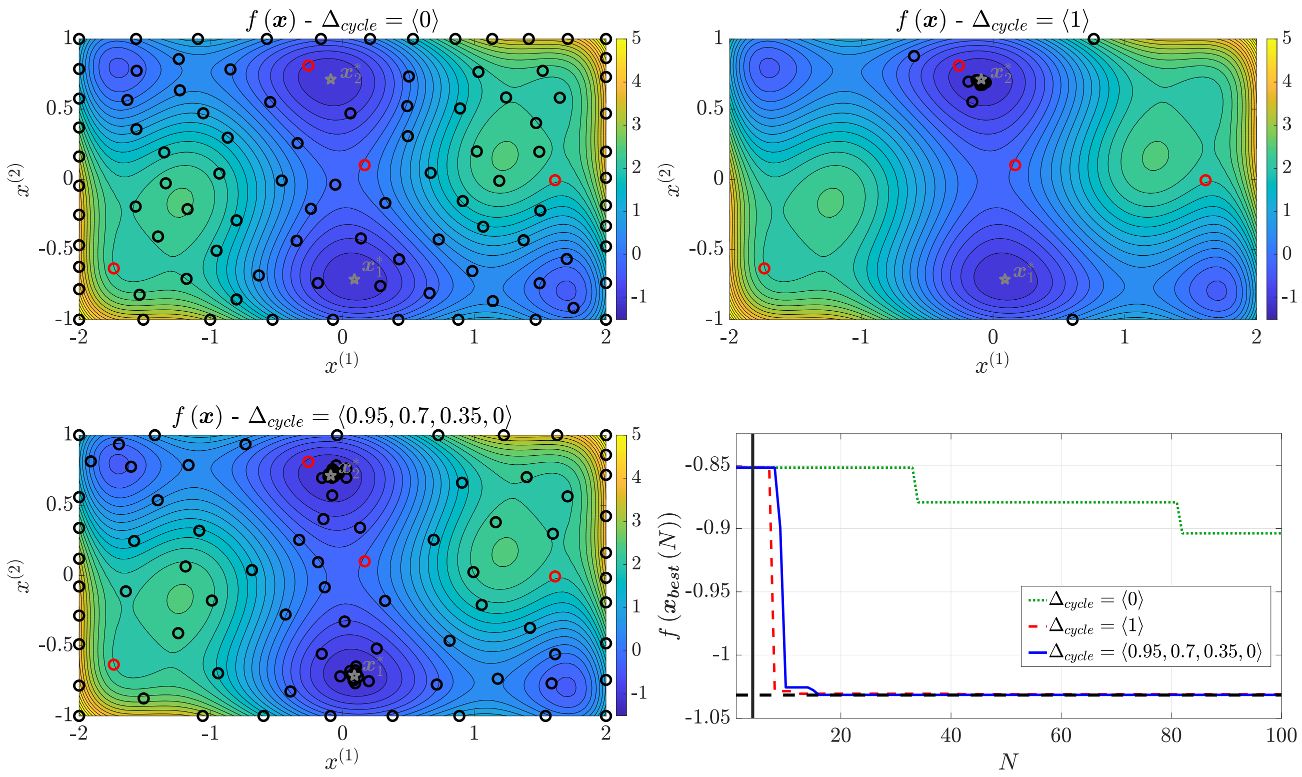
Example 5.2: Performances for different Δ_{cycle}'s

Consider the `camel six humps` [62] function defined in Example 4.2. We perform its minimization over the constraint set $\Omega = \{\boldsymbol{x} : \boldsymbol{l} \le \boldsymbol{x} \le \boldsymbol{u}\}, \boldsymbol{l} = \begin{bmatrix}-2 & -1\end{bmatrix}^\top, \boldsymbol{u} = \begin{bmatrix}2 & 1\end{bmatrix}^\top$, using `GLIS-r` [108] and `GLISp-r` [109] with the following hyper-parameters:

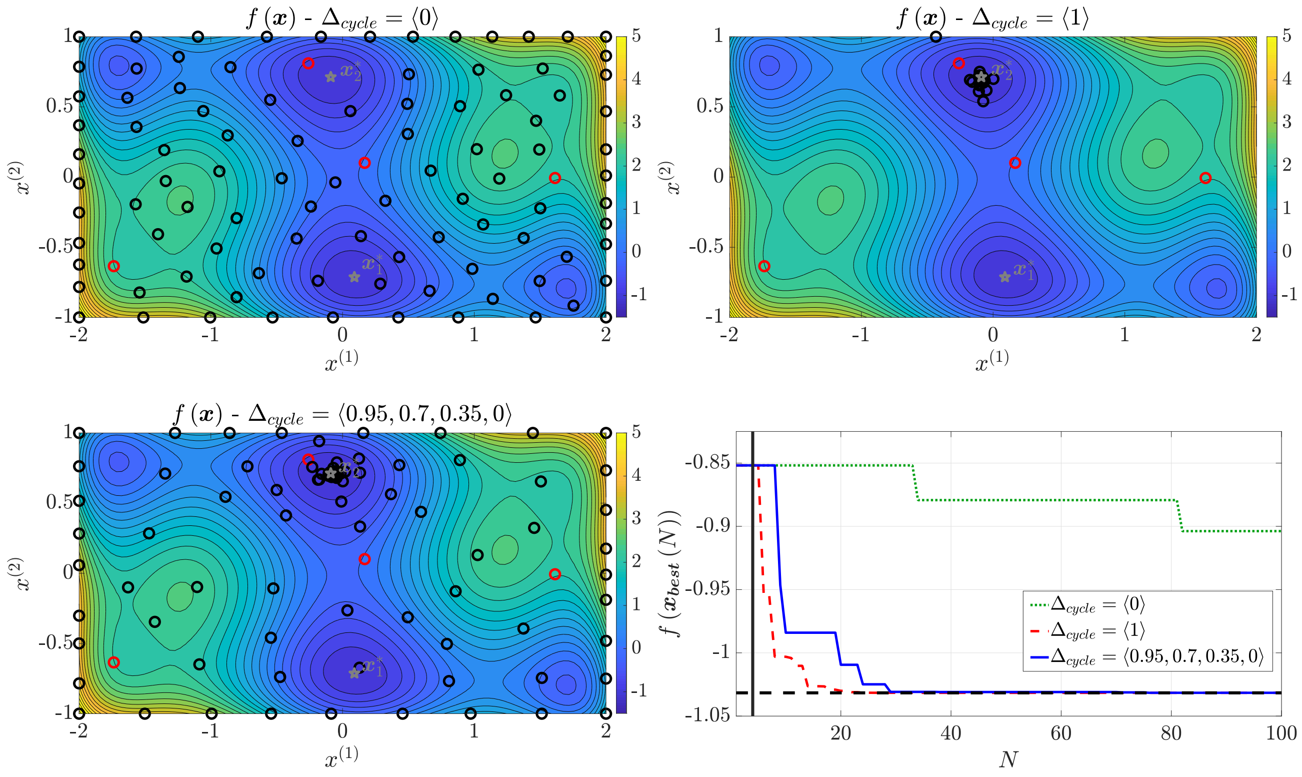
- Number of initial samples $N_{init} = 4$,
- Budget $N_{max} = 100$,
- Hyper-parameters for the surrogate model $\hat{f}_N(\boldsymbol{x})$ in (4.1):
  - (BBO) $\epsilon_f = 0.5378$, $\varphi_f(\cdot)$ inverse quadratic, $\epsilon_{SVD} = 10^{-6}$,
  - (PBO) $\epsilon_f = 1$, $\varphi_f(\cdot)$ inverse quadratic, $\lambda_f = 10^{-6}$, $\sigma_\pi = 10^{-2}$.

- We compare three different cycling sets:
  - $\Delta_{cycle} = \langle 0 \rangle$ (pure exploration),
  - $\Delta_{cycle} = \langle 1 \rangle$ (pure exploitation),
  - $\Delta_{cycle} = \langle 0.95, 0.7, 0.35, 0 \rangle$. The rationale behind this cycling set is to "equally" alternate between exploitation and exploration, starting from higher δ's so that, at the first iterations of GLIS-r [108] and GLISp-r [109], the surrogate models are exploited to find promising regions of Ω. Then, we switch to exploration in order to avoid missing the other local/global minimizers of the GOP (2.1).
- Number of clusters for $\mathcal{X}_{aug}$: $K_{aug} = 5$,
- No recalibration of $\hat{f}_N(\boldsymbol{x})$ in (4.1), i.e. $\mathcal{K}_{R_f} = \emptyset, \mathcal{M}_f = \emptyset$.

Problem (5.13) is solved by the PSWARM [72] algorithm (see Section 1.2.5). The considered GOP (2.1) has two global solutions: $\boldsymbol{x}_1^* = \begin{bmatrix} 0.0898 & -0.7126 \end{bmatrix}^\top$ and $\boldsymbol{x}_2^* = -\boldsymbol{x}_1^*$. Figure 24 depicts the results obtained by GLIS-r [108] and GLISp-r [109] for the different Δ_{cycles}'s. Notice how the cycling set $\Delta_{cycle} = \langle 0.95, 0.7, 0.35, 0 \rangle$ is able to (approximately) locate both global minimizers of the considered GOP (2.1). Instead, at least in this Example, the pure exploitation approach is able to find only $\boldsymbol{x}_2^*$ but converges faster than the other two strategies. *In general, setting $\Delta_{cycle} = \langle 1 \rangle$ for GLIS-r [108] and GLISp-r [109] does not guarantee that the procedures converge to a global minimizer of the GOP* (2.1) (see Section 5.3.1). Lastly, the pure exploration approach is quite slower than the other two choices of Δ_{cycle}. That is to be expected, although exploration is a "necessary evil" for ensuring the convergence of any global optimization procedure.



(a) Performances of `GLIS-r` [108] (black-box optimization).



(b) Performances of `GLISp-r` [109] (preference-based optimization).

Figure 24: Comparison of the performances achieved by `GLIS-r` [108] and `GLISp-r` [109] when equipped with different cycling sets, as described in Example 5.2. In the level curves graphs, the red circles are the N_{init} initial samples, whereas the ones in black are obtained by solving Problem (5.13). The stars in grey are the global minimizers of the GOP (2.1). We also report the convergence plots of `GLIS-r` [108] and `GLISp-r` [109] for the different Δ_{cycle}'s. Therein, the dashed black-line is the global minima of the GOP (2.1). Instead, the black vertical line denotes N_{init}.

Algorithm 14: GLIS-r [108] and GLISp-r [109]

Input: (i) A-priori known constraint set Ω of the GOP (2.1); (ii) Number of initial samples $N_{init} \in \mathbb{N}$; (iii) Budget $N_{max} \in \mathbb{N}, N_{max} > N_{init}$; (iv) Hyper-parameters for the surrogate model $\hat{f}_N(\boldsymbol{x})$ in (4.1): shape parameter $\epsilon_f \in \mathbb{R}_{>0}$ and radial function $\varphi_f(\cdot)$ (BBO and PBO), threshold $\epsilon_{SVD} \in \mathbb{R}_{>0}$ (BBO), regularization parameter $\lambda_f \in \mathbb{R}_{\geq 0}$ and tolerance $\sigma_\pi \in \mathbb{R}_{>0}$ (PBO); (v) *Cycling set Δ_{cycle} in (5.15) for the acquisition function $a_N(\boldsymbol{x})$ in (5.12)*; (vi) *Number of clusters $K_{aug} \in \mathbb{N}$ for the augmented sample set $\mathcal{X}_{aug}$ generated by Algorithm 13*; (vii) Set of models $\mathcal{M}_f$ in (4.30) for the recalibration of the surrogate model $\hat{f}_N(\boldsymbol{x})$ in (4.1); (viii) Set of indexes for the recalibration of the surrogate model $\hat{f}_N(\boldsymbol{x})$ in (4.1), $\mathcal{K}_{R_f} \subseteq \{1, \dots, N_{max} - N_{init}\}$, and folds ratio $R_f \in [0, 1]$.

Output: (i) Best cost obtained by the procedure $y_{best}(N_{max})$ (only for BBO); (ii) Best sample obtained by the procedure $\boldsymbol{x_{best}}(N_{max})$.

```
 1: Rescale the GOP (2.1) as proposed in Section 4.4.1, obtaining Problem (4.27)
 2: Generate a set $\mathcal{X}$ in (2.9) of $N_{init}$ starting points using a LHD (see Section 2.4)
 3: Evaluate the samples in $\mathcal{X}$ either by measuring the values of $f(\cdot)$, obtaining the set $\mathcal{Y}$ in (2.10) (BBO), or
    by querying the decision-maker as in Algorithm 7, obtaining the sets $\mathcal{B}$ (3.9) and $\mathcal{S}$ (3.10), as well as the
    best candidate $\boldsymbol{x_{best}}(N_{init})$
 4: Set $N = N_{init}$ (and $M = |\mathcal{B}|$ for PBO)
 5: Set $\delta = \delta_0 \in \Delta_{cycle}$ and $j = 0$
 6: for $k = 1, 2, \dots, N_{max} - N_{init}$ do
 7:     if $k \in \mathcal{K}_{R_f}$ then recalibrate the surrogate model $\hat{f}_N(\boldsymbol{x})$ in (4.1) as in Algorithm 8
 8:     Build the surrogate model for $f(\boldsymbol{x})$ using the information at hand: in BBO, find $\boldsymbol{\beta}_f$ as in (4.4) whereas,
        for PBO, select $\boldsymbol{\beta}_f$ as the solution of Problem (4.10)
 9:     Generate the augmented sample set $\mathcal{X}_{aug}$ through Algorithm 13
10:     Look for the next candidate sample $\boldsymbol{x}_{N+1}$ by solving Problem (5.13) with $a_N(\boldsymbol{x})$ in (5.12)
11:     Evaluate the new candidate sample, obtaining either $y_{N+1} = f(\boldsymbol{x}_{N+1})$ (in BBO) or $b_{M+1} =$
        $\pi_{\gtrsim}(\boldsymbol{x}_{N+1}, \boldsymbol{x_{best}}(N))$ (in PBO)
12:     Update the set of samples $\mathcal{X}$ and either the set of measures of $f(\cdot)$, $\mathcal{Y}$ (BBO), or the preference
        information in the sets $\mathcal{B}$ and $\mathcal{S}$ (PBO)
13:     Check if $\boldsymbol{x}_{N+1}$ improves upon $\boldsymbol{x_{best}}(N)$, as highlighted by the flag $hasImproved$ from Algorithm 9
14:     if $hasImproved$ then
15:         Set $\boldsymbol{x_{best}}(N+1) = \boldsymbol{x}_{N+1}$
16:     else
17:         Keep $\boldsymbol{x_{best}}(N+1) = \boldsymbol{x_{best}}(N)$
18:         Set $\delta = \delta_{(j+1) mod N_{cycle}} \in \Delta_{cycle}$ (greedy $\delta$-cycling) and $j = j + 1$
19:     Set $N = N + 1$ (and $M = M + 1$ for PBO)
```

5.3.1 Convergence

In this Section, we prove the convergence of the GLIS-r [108] and GLISp-r [109] algorithms to the global minima of the GOP (2.1). We point out that, *at the moment, no proof of convergence is available for GLIS [10] and GLISp [11].* Before proceeding, recall that, in BBO, we assume the measures of the black-box cost function to be noiseless (see Assumption 2.4). The previous Assumption is needed to prove the asymptotic convergence to the *exact* global minima of the GOP (2.1) (although it does not preclude the denseness of the sequences of iterates produced by GLIS-r [108] and GLISp-r [109], required by Theorem 1.2).

First of all, we need to ensure that the GOP (2.1) admits at least a solution. Theorem 1.1 and Proposition 3.1 give us sufficient conditions to guarantee that the set of global minimizers of the GOP (2.1) is nonempty, i.e. $\mathcal{X}^* \neq \emptyset$. To summarize:

1. The constraint set Ω of the GOP (2.1) must be compact,
2. In BBO, the black-box cost function $f(\boldsymbol{x})$ must be continuous. Instead, in PBO, the preference relation $\gtrsim$ on Ω of the human decision-maker must be continuous (see Definition 3.4) and the DM must be rational (see Definition 3.1). Recall that, by Theorem 3.1, if the conditions in Proposition 3.1 hold, then $\gtrsim$ can be represented by a continuous utility function $u_{\gtrsim}(\boldsymbol{x})$. In turn, this makes the cost function of the GOP (2.1) in PBO, which is the scoring function $f(\boldsymbol{x}) = -u_{\gtrsim}(\boldsymbol{x})$ of the DM, continuous (cf. Remark 3.2).

If the GOP (2.1) admits a solution, then any global optimization algorithm that produces a sequence of iterates, $\langle \boldsymbol{x}_i \rangle_{i \geq 1}$, that is dense in Ω is guaranteed to (ultimately) find it, see Theorem 1.2. For the remainder of this Chapter, we use the following notation.

Notation and conventions. Consider the sequence of iterates $\langle \boldsymbol{x}_i \rangle_{i \geq 1}$ produced by a global optimization algorithm. We define:

- $\mathcal{X}_\infty$ as the set containing all the elements of $\langle \boldsymbol{x}_i \rangle_{i \geq 1}$,
- $\mathcal{X}_k \subseteq \mathcal{X}_\infty$ as the set containing all the elements of $\langle \boldsymbol{x}_i \rangle_{i=1}^{k}$, which is a subsequence of $\langle \boldsymbol{x}_i \rangle_{i \geq 1}$ composed of its first $k \in \mathbb{N}$ entries.

The following Theorem gives us a sufficient condition that ensures the denseness of $\mathcal{X}_\infty$.

Theorem 5.1: A sufficient condition for the denseness of $\mathcal{X}_\infty$ [115]

Let Ω be a compact subset of $\mathbb{R}^n$ and let $\langle \boldsymbol{x}_i \rangle_{i \geq 1}$ be the sequence of iterates generated by an algorithm A *(when run indefinitely). Suppose that there exists a strictly increasing sequence of positive integers $\langle i_t \rangle_{t \geq 1}, i_t \in \mathbb{N}$, such that $\langle \boldsymbol{x}_i \rangle_{i \geq 1}$ satisfies the following condition for some $\alpha \in (0, 1]$:*

$$\min_{1 \leq i \leq i_t - 1} \left\| \boldsymbol{x}_{i_t} - \boldsymbol{x}_i \right\|_2 \geq \alpha \cdot d_\Omega \left(\mathcal{X}_{i_t - 1} \right), \quad \forall t \in \mathbb{N}, \tag{5.17}$$

where:

$$d_\Omega \left(\mathcal{X}_{i_t - 1} \right) = \max_{\boldsymbol{x} \in \Omega} \min_{1 \leq i \leq i_t - 1} \left\| \boldsymbol{x} - \boldsymbol{x}_i \right\|_2 . \tag{5.18}$$

Then, $\mathcal{X}_\infty$ generated by A *is dense in Ω. Hence,* A *converges to the global minimum of any continuous function $f : \mathbb{R}^n \to \mathbb{R}$ over Ω.*

Proof. Define the sequence $\langle d_t \rangle_{t \geq 1}$ of minimum distances between the points $\boldsymbol{x}_{i_t}$ (indexed by the elements of $\langle i_t \rangle_{t \geq 1}$) and the samples in $\mathcal{X}_{i_t - 1}$ (which does not include $\boldsymbol{x}_{i_t}$) as:

$$d_t = \min_{1 \leq i \leq i_t - 1} \left\| \boldsymbol{x}_{i_t} - \boldsymbol{x}_i \right\|_2 . \tag{5.19}$$

Now, suppose that $\mathcal{X}_\infty$ is not dense in Ω. Then, there exists a point $\tilde{\boldsymbol{x}} \in \Omega$ and a constant $\epsilon \in \mathbb{R}_{>0}$ such that the open ball centered at $\tilde{\boldsymbol{x}} \in \Omega$ and with radius ϵ (namely $\mathcal{B}\left(\tilde{\boldsymbol{x}}; \epsilon\right)$) does not contain any element of $\mathcal{X}_\infty$, i.e.:

$$\|\tilde{\boldsymbol{x}} - \boldsymbol{x}_i\|_2 \geq \epsilon, \quad \forall \boldsymbol{x}_i \in \mathcal{X}_\infty. \tag{5.20}$$

From (5.17), (5.19) and (5.20) we have that:

$$d_t \geq \alpha \cdot d_\Omega \left(\mathcal{X}_{i_t - 1}\right) \geq \alpha \cdot \epsilon > 0, \quad \forall t \in \mathbb{N}. \tag{5.21}$$

Let $\tilde{\epsilon} = \alpha \cdot \epsilon$. Since $d_t \geq \tilde{\epsilon}, \forall t \in \mathbb{N}$, we have that:

$$\left\| \boldsymbol{x}_{i_{t'}} - \boldsymbol{x}_{i_{t''}} \right\|_2 \geq \tilde{\epsilon}, \quad \text{for any } t' > t''. \tag{5.22}$$

Ω is a compact subset of $\mathbb{R}^n$ and hence it is bounded (see Theorem A.2). Therefore, we can build a hypercube Ω_H which contains Ω, i.e. $\Omega \subseteq \Omega_H$. Define the side length of Ω_H as $\frac{r \cdot \tilde{\epsilon}}{2 \cdot \sqrt{n}}$ for some positive integer $r \in \mathbb{N}$. Partition Ω_H into r^n equally-sized hypercubes,

$$\Omega_H^{(1)}, \ldots, \Omega_H^{(r^n)}, \tag{5.23}$$

where each $\Omega_H^{(h)}, h = 1, \ldots, r^n$, has side length $\frac{\tilde{\epsilon}}{2 \cdot \sqrt{n}} < \tilde{\epsilon}$ and the length of its main diagonal is $\frac{\tilde{\epsilon}}{2}$. Clearly, $\tilde{\boldsymbol{x}}$ must lie in one of the hypercubes in (5.23). Now, from (5.22), any two elements of the sequence $\langle \boldsymbol{x}_{i_t} \rangle_{t \geq 1}$, say $\boldsymbol{x}_{i_{t'}}$ and $\boldsymbol{x}_{i_{t''}}$ with $t' > t''$, must belong to different hypercubes in (5.23), since $\left\| \boldsymbol{x}_{i_{t'}} - \boldsymbol{x}_{i_{t''}} \right\|_2$ is greater than the length of the main diagonal of any $\Omega_H^{(h)}, h = 1, \ldots, r^n$. But this is a contradiction because $\langle \boldsymbol{x}_{i_t} \rangle_{t \geq 1}$ is an infinite sequence of distinct points (see (5.22)) whereas $\Omega_H^{(1)}, \ldots, \Omega_H^{(r^n)}$ is only a finite partition of Ω. Consequently, $\exists \boldsymbol{x}_{i_t}$ in the sequence $\langle \boldsymbol{x}_{i_t} \rangle_{t \geq 1}$ such that $\boldsymbol{x}_{i_t} \in \mathcal{B}\left(\tilde{\boldsymbol{x}}; \epsilon\right)$. Thus, $\mathcal{X}_\infty$, which contains all the elements of the sequence $\langle \boldsymbol{x}_{i_t} \rangle_{t \geq 1}$, must be dense in Ω. Finally, by Theorem 1.2, algorithm A converges to the global minima of the GOP (2.1). □

Theorem 5.1 has been used to prove the convergence of the CORS [115] algorithm (see Section 2.6). We could interpret the previous Theorem as follows: in order to guarantee the convergence of an algorithm A, we must ensure that, infinitely often, it produces samples $\boldsymbol{x}_{i_t}$ (indexed by the elements of $\langle i_t \rangle_{t \geq 1}$) that are at least $\epsilon = \alpha \cdot d_\Omega \left(\mathcal{X}_{i_t - 1}\right) > 0$ away from the previously evaluated ones. Intuitively, $d_\Omega \left(\mathcal{X}_{i_t - 1}\right)$ in (5.18) is the maximum distance between any point contained inside the constraint set Ω and its closest sample in $\mathcal{X}_{i_t - 1}$.

Now, consider a compact constraint set Ω and a continuous acquisition function $a_N(\boldsymbol{x})$. Coherently with most surrogate-based methods (see Section 2.6), all elements of the sequence $\langle \boldsymbol{x}_{i_t} \rangle_{t \geq 1}$ are obtained by solving:

$$\begin{aligned} \boldsymbol{x}_{i_t} &= \arg \min_{\boldsymbol{x}} a_{i_t - 1}(\boldsymbol{x}) \\ \text{s.t.} &\quad \boldsymbol{x} \in \Omega, \end{aligned}$$

hence $\boldsymbol{x}_{i_t} \in \Omega, \forall t \in \mathbb{N}$. Therefore, we could extend the condition in (5.17) to:

$$\alpha \cdot d_\Omega\left(\mathcal{X}_{i_t - 1}\right) \leq \min_{1 \leq i \leq i_t - 1} \left\| \boldsymbol{x}_{i_t} - \boldsymbol{x}_i \right\|_2 \leq d_\Omega\left(\mathcal{X}_{i_t - 1}\right), \quad \forall t \in \mathbb{N}. \tag{5.24}$$

We are now ready to state the convergence Theorem for `GLIS-r` [108] and `GLISp-r` [109].

Theorem 5.2: Convergence of `GLIS-r` [108] and `GLISp-r` [109]

Let $\Omega \subset \mathbb{R}^n$ be a compact set and either:

- *$f : \mathbb{R}^n \to \mathbb{R}$ be a continuous function (BBO) or*
- *$\gtrsim$ be a continuous preference relation on Ω of a rational (as in Definition 3.1) human decision-maker (PBO).*

Then, provided that $\exists \delta_j \in \Delta_{cycle}$ in (5.15) such that $\delta_j = 0$ and $N_{max} \to \infty$, Algorithm 14 converges to the global minima of the GOP (2.1) for any choice of its remaining hyper-parameters[a]*.*

[a]Formally, we should also ensure that the surrogate model $\hat{f}_N(\boldsymbol{x})$ in (4.1) is continuous. However, that is the case for any of the radial basis functions reported in Definition 2.6.

Proof. Compactness of Ω, continuity of $f(\boldsymbol{x})$ or $\gtrsim$ and rationality of the DM are conditions that ensure the existence of a solution for the GOP (2.1).
Consider the sequence of iterates $\langle \boldsymbol{x}_i \rangle_{i \geq 1}$ produced by Algorithm 14. The first $N_{init} \in \mathbb{N}$ elements of $\langle \boldsymbol{x}_i \rangle_{i \geq 1}$ are obtained by the LHD. Instead, each $\boldsymbol{x}_i \in \mathcal{X}_\infty, i > N_{init}$, is selected as the solution of Problem (5.13).
Now, suppose that δ in (5.12) is cycled regardless of the improvement that the new candidate samples might bring (non-greedy cycling). Denote the exploration-exploitation trade-off weight at iteration k of Algorithm 14 as $\delta(k)$ and assume that $\delta(k) = \delta_j \in \Delta_{cycle}$. Then, the cycling is performed as:

$$\delta(k+1) = \delta_{(j+1) \bmod N_{cycle}}, \quad \forall k \in \mathbb{N}, \tag{5.25}$$

instead of (5.16). Without loss of generality, suppose that Δ_{cycle} in (5.15) is defined as:

$$\delta_j \neq 0, \forall j = 0, \ldots, N_{cycle} - 2, \text{ and } \delta_{N_{cycle}-1} = 0.$$

Then, every N_{cycle} iterations Algorithm 14 looks for a new candidate sample by minimizing the (min-max rescaled) IDW distance function in (4.17), regardless of the surrogate model in (4.1), see $a_N(\boldsymbol{x})$ in (5.12) and Problem (5.13). We define the following strictly increasing sequence of positive integers:

$$\langle i_{t'} \rangle_{t' \geq 1} = \langle N_{init} + t' \cdot N_{cycle} \rangle_{t' \geq 1}, \tag{5.26}$$

that is such that:

$$\begin{aligned} \boldsymbol{x}_{i_{t'}} = \arg \min_{\boldsymbol{x}} \bar{z}_{i_{t'}-1}\left(\boldsymbol{x}; \mathcal{X}_{aug}\right), \quad \forall t' \in \mathbb{N} \\ \text{s.t.} \quad \boldsymbol{x} \in \Omega. \end{aligned} \tag{5.27}$$

Problem (5.27) is equivalent to:

$$\begin{aligned} \boldsymbol{x}_{i_{t'}} = \arg \min_{\boldsymbol{x}} z_{i_{t'}-1}(\boldsymbol{x}), \quad \forall t' \in \mathbb{N} \\ \text{s.t.} \quad \boldsymbol{x} \in \Omega, \end{aligned} \tag{5.28}$$

since scaling and shifting the IDW distance function in (4.17) does not change its minimizers [100]. Now, recall from Proposition 5.1 that each $\boldsymbol{x}_i \in \mathcal{X}_{i_{t'}-1}$ is a global maximizer of Problem (5.28). Furthermore, $z_{i_{t'}-1}(\boldsymbol{x})$ in (4.17) is differentiable everywhere and thus continuous (see Proposition 4.4). Then, by the Extreme Value Theorem 1.1, Problem (5.28) admits at least a solution. Hence, we can conclude that:

$$\boldsymbol{x}_{i_{t'}} \notin \mathcal{X}_{i_{t'}-1} \implies \exists \epsilon' \in \mathbb{R}_{>0} \text{ such that } \min_{1 \leq i \leq i_{t'}-1} \left\| \boldsymbol{x}_{i_{t'}} - \boldsymbol{x}_i \right\|_2 \geq \epsilon', \quad \forall t' \in \mathbb{N}. \tag{5.29}$$

Clearly, from (5.24), we have that:

$$\epsilon' \leq d_{\Omega}\left(\mathcal{X}_{i_{t'}-1}\right), \quad \forall t' \in \mathbb{N}. \tag{5.30}$$

By combining (5.29) and (5.30), we get:

$$0 < \epsilon' \leq d_{\Omega}\left(\mathcal{X}_{i_{t'}-1}\right), \quad \forall t' \in \mathbb{N}. \tag{5.31}$$

Therefore, $\exists \alpha' \in (0, 1]$ such that $\epsilon' = \alpha' \cdot d_{\Omega}\left(\mathcal{X}_{i_{t'}-1}\right), \forall t' \in \mathbb{N}$. The condition (5.17) of Theorem 5.1 is satisfied and thus Algorithm 14 with δ cycled as in (5.25) converges to the global minima of the GOP (2.1).

Next, consider the greedy δ-cycling strategy proposed in Section 5.2.3 and for an arbitrary choice of Δ_{cycle} in (5.15). Let us examine the case in (5.16) when δ is kept unchanged from an iteration of Algorithm 14 to the other. Denote as $\langle i_{t''} \rangle_{t''=1}^{t''_{max}}, t''_{max} \in \mathbb{N}$, the sequence of indexes of those samples that improve upon the current best candidate, resulting in no change in the exploration-exploitation trade-off weight. We have that:

$$y_{i_{t''}} < y_{best}\left(i_{t''}-1\right) \text{(BBO) or } \boldsymbol{x}_{i_{t''}} \succ \boldsymbol{x_{best}}\left(i_{t''}-1\right) \text{(PBO)}, \quad \forall t'' : 1 \leq t'' \leq t''_{max}. \tag{5.32}$$

Clearly, $\boldsymbol{x}_{i_{t''}} \notin \mathcal{X}_{i_{t''}-1}$ since it either achieves a lower value of $f(\cdot)$ or it is strictly preferred to all the other samples. Thus, we could define a positive constant $\epsilon'' \in \mathbb{R}_{>0}$ as in (5.29). The caveat is that the sequence $\langle i_{t''} \rangle_{t''=1}^{t''_{max}}$ might be finite if Δ_{cycle} in (5.15) does not include a zero entry. That is because we have no guarantee that the solution of:

$$\begin{aligned} \boldsymbol{x}_{N+1} = \arg\min_{\boldsymbol{x}} a_N(\boldsymbol{x}), \quad & \forall N : N \neq i_{t''}-1, 1 \leq t'' \leq t''_{max} \\ \text{s.t.} \quad \boldsymbol{x} \in \Omega, & \end{aligned} \tag{5.33}$$

with $a_N(\boldsymbol{x})$ defined as in (5.12), is not already present in $\mathcal{X}_N$. Also note that Problem (5.33) always admits a solution since $a_N(\boldsymbol{x})$ in (5.12) is continuous and Ω is compact. Alternatively speaking, if $\nexists \delta_j \in \Delta_{cycle}$ such that $\delta_j = 0$, then we have no guarantee that Algorithm 14 improves its current best candidate infinitely often. Hence, the result in Theorem 5.1 does not apply for $\langle i_{t''} \rangle_{t''=1}^{t''_{max}}$, since the sequence is finite[2].

Finally, let us consider the greedy δ-cycling strategy in (5.16) as whole and assume, as in Theorem 5.2, that $\exists \delta_j \in \Delta_{cycle}$ in (5.15) such that $\delta_j = 0$. We can build a strictly increasing sequence of positive integers $\langle i_t \rangle_{t\geq 1}$ by merging:

- The elements of the sequence $\langle i_{t''} \rangle_{t''=1}^{t''_{max}}$, which are the indexes of those samples that improve upon the best candidates $\boldsymbol{x_{best}}\left(i_{t''}-1\right), \forall t'' : 1 \leq t'' \leq t''_{max}$;

- The elements of the sequence $\langle i_{t'} \rangle_{t'\geq 1}$, which constitute the indexes of those samples found by solving the pure exploration Problem (5.28). Note that, unless Algorithm 14 always improves upon its current best candidate (in which case $\langle i_{t''} \rangle_{t''=1}^{t''_{max}}$ is actually infinite, and hence $\mathcal{X}_\infty$ is dense in Ω), Problem (5.13) is solved with $\delta = 0$ for the acquisition function $a_N(\boldsymbol{x})$ in (5.12) infinitely often, although not necessarily every N_{cycle} iterations as in (5.26).

Hence, we can select a positive constant $\epsilon \in \mathbb{R}_{>0}$ as $\epsilon = \min\{\epsilon', \epsilon''\}$ which, analogously to (5.31), is such that:

$$0 < \epsilon \leq d_\Omega\left(\mathcal{X}_{i_t-1}\right), \quad \forall t \in \mathbb{N}. \tag{5.34}$$

[2]Note that finiteness of $\langle i_{t''} \rangle_{t''=1}^{t''_{max}}$ does not necessarily preclude the global convergence of Algorithm 14 for all possible GOPs (2.1). For example, if $f(\boldsymbol{x})$ is a constant function then, after we evaluate the first sample $\boldsymbol{x}_1$, any other point brings no improvement. However, finiteness of $\langle i_{t''} \rangle_{t''=1}^{t''_{max}}$ implies that $\mathcal{X}_\infty$ is not dense in Ω, hence we cannot guarantee the converge of Algorithm 14 for any continuous $f(\boldsymbol{x})$ over Ω.

Therefore, $\exists \alpha \in (0,1]$ such that $\epsilon = \alpha \cdot d_\Omega\left(\mathcal{X}_{i_t-1}\right), \forall t \in \mathbb{N}$. The condition (5.17) of Theorem 5.1 is satisfied and thus Algorithm 14 with δ in (5.12) cycled following the greedy δ-cycling strategy in (5.16) converges to the global minima of the GOP (2.1). □

We end this Section with some concluding Remarks.

Remark 5.4 (A note on the convergence in the preference-based framework)**.** *Most preference-based response surface techniques, such as the ones reviewed in Section 3.5, do not address their convergence to the global minima of the GOP* (2.1)*. In this book, we have shown that, by leveraging some results from the utility theory framework (see Section 3.1 and Section 3.2), we can find sufficient conditions on the preference relation of the human decision-maker ($\gtrsim$ on Ω) that guarantee the existence of a solution for the GOP* (2.1) *and allow us to analyze the convergence of any PBO procedure as we would in BBO. In particular, if the DM is rational and $\gtrsim$ on Ω is continuous, then denseness of the sequence of iterates $\langle \boldsymbol{x}_i \rangle_{i \geq 1}$ produced by a preference-based optimization algorithm* `A` *guarantees the convergence of* `A` *to the $\gtrsim$-maxima of Ω.*

Remark 5.5 (Choice of the cycling set)**.** *Theorem 5.2 guarantees that, under some hypotheses,* `GLIS-r` *[108] and* `GLISp-r` *[109] converge to the global minima of the GOP* (2.1)*, however it does not give any indication on their convergence rates. In particular, if Δ_{cycle} is actually $\langle 0 \rangle$, Algorithm 14 amounts to performing an exhaustive search without considering the information on $f(\boldsymbol{x})$ brought by the samples in $\mathcal{X}$* (2.9)*, which is quite inefficient. Therefore, it is best to include some δ_j's in Δ_{cycle} that allow the surrogate model to be taken into consideration. For this reason, we suggest including terms that are well spread within the* $[0,1]$ *range, including a zero entry to ensure the result in Theorem 5.2. Intuitively, the rates of convergence will be dependent on how well $\hat{f}_N(\boldsymbol{x})$ in* (4.1) *approximates $f(\boldsymbol{x})$ as well as on the choice of Δ_{cycle} in* (5.15).

5.4 A general surrogate-based scheme for unconstrained BBO and PBO

In this Section, we generalize Algorithm 14 so that it can handle any (continuous) surrogate model as well as any (proper) exploration function. In the unconstrained black-box optimization literature, there exist some surrogate-based methods, such as `MSRS` [116] and `CORS` [115], which make no specific assumptions on $\hat{f}_N(\boldsymbol{x})$ as to let the user select the one that is best suited for the application at hand. There exist alternative surrogate models to the ones based on RBFs or $\mathcal{GP}$s, reviewed in Chapter 2 and Chapter 3. In black-box optimization, polynomials, Support Vector Machines (SVMs) for regression and neural networks have also been used to approximate the black-box cost function of the GOP (2.1)

[65, 149]. Instead, in surrogate-based PBO, fewer surrogate models have been tried (see Section 3.4), although alternatives exist. For example, in [141], the author proposes a neural network architecture to estimate the utility function of a human decision-maker.

In what follows, we define a general surrogate-based scheme which we refer to as generalized Metric Response Surface (gMRS [108]), since it can be seen as an extension of the MSRS [116] scheme that can handle both black-box and preference-based optimization. gMRS [108] adopts the same infill sampling criterion proposed in Section 5.2 but takes into account that different surrogate models $\hat{f}_N(\boldsymbol{x})$ and exploration functions $z_N(\boldsymbol{x})$ can be employed for the definition of the acquisition function in (5.12). Similarly, the exploration-exploitation dilemma is addressed by the greedy δ-cycling strategy in Section 5.2.3. Here, we give sufficient conditions on $\hat{f}_N(\boldsymbol{x})$ and $z_N(\boldsymbol{x})$ which, combined with the inclusion of a zero entry in Δ_{cycle} in (5.15), guarantee the global convergence of gMRS [108]. In practice, GLIS-r [108] and GLISp-r [109] are two implementations of the gMRS [108] scheme that use the surrogate model in (4.1) and the exploration function in (4.17).

5.4.1 Infill sampling criterion

As previously pointed out, gMRS [108] relies on the acquisition function $a_N(\boldsymbol{x})$ in (5.12) to drive the search for new candidate samples, although $\hat{f}_N(\boldsymbol{x})$ and $z_N(\boldsymbol{x})$ need not be, respectively, the ones in (4.1) and (4.17). Moreover, the augmented sample set $\mathcal{X}_{aug}$ in (5.12) can be generated differently from Algorithm 13, since the latter is tailored for the IDW distance function in (4.17). Formally:

Definition 5.3: Acquisition function for gMRS [108]. *The aforementioned scheme adopts an infill sampling criterion that is based on the following acquisition function $a_N : \mathbb{R}^n \to \mathbb{R}$:*

$$a_N(\boldsymbol{x}) = \delta \cdot \bar{\hat{f}}_N\left(\boldsymbol{x}; \mathcal{X}_{aug}\right) + (1-\delta) \cdot \bar{z}_N\left(\boldsymbol{x}; \mathcal{X}_{aug}\right), \tag{5.35}$$

where:

- *$\hat{f}_N : \mathbb{R}^n \to \mathbb{R}$ is a <u>continuous</u> surrogate model,*
- *$z_N : \mathbb{R}^n \to \mathbb{R}$ is a <u>proper</u> exploration function (see Section 5.4.2),*
- *The latter two functions have been rescaled using min-max normalization (see Definition 5.1),*

- $\mathcal{X}_{aug} = \left\{\boldsymbol{x}_{\boldsymbol{aug}_i} : i = 1, \dots, N_{aug}, \boldsymbol{x}_{\boldsymbol{aug}_i} \in \Omega\right\}, N_{aug} \in \mathbb{N}$, *is the augmented sample set, which needs to be defined so that:*

$$\hat{f}_{N_{min}}\left(\mathcal{X}_{aug}\right) \approx \min_{\boldsymbol{x}\in\Omega} \hat{f}_N(\boldsymbol{x}), \tag{5.36a}$$

$$\hat{f}_{N_{max}}\left(\mathcal{X}_{aug}\right) \approx \max_{\boldsymbol{x}\in\Omega} \hat{f}_N(\boldsymbol{x}), \tag{5.36b}$$

$$z_{N_{min}}\left(\mathcal{X}_{aug}\right) \approx \min_{\boldsymbol{x}\in\Omega} z_N(\boldsymbol{x}), \tag{5.36c}$$

$$z_{N_{max}}\left(\mathcal{X}_{aug}\right) \approx \max_{\boldsymbol{x}\in\Omega} z_N(\boldsymbol{x}). \tag{5.36d}$$

- $\delta \in [0, 1]$ *is the exploration-exploitation trade-off weight.*

Analogously to GLIS-r [108] and GLISp-r [109], gMRS [108] looks for new candidate samples by minimizing the acquisition function $a_N(\boldsymbol{x})$ in (5.35), i.e.:

$$\begin{aligned} \boldsymbol{x}_{N+1} = \arg\min_{\boldsymbol{x}} & \; a_N(\boldsymbol{x}) \\ \text{s.t.} & \quad \boldsymbol{x} \in \Omega. \end{aligned} \tag{5.37}$$

The exploration-exploitation trade-off weight δ in (5.35) is cycled following the greedy δ-cycling paradigm described in Section 5.2.3. Concerning the augmented sample set, an adequate choice of $\mathcal{X}_{aug}$ for $a_N(\boldsymbol{x})$ in (5.35) makes the surrogate model $\hat{f}_N(\boldsymbol{x})$ and the exploration function $z_N(\boldsymbol{x})$ comparable, as we have seen in Section 5.2. Alternatively to Algorithm 13, $\mathcal{X}_{aug}$ can be selected as follows:

- Set $\mathcal{X}_{aug} = \mathcal{X}$. This is not recommended since each $\boldsymbol{x}_i \in \mathcal{X}$ in (2.9) might maximize the exploration function $z_N(\boldsymbol{x})$. That is the case for the IDW distance function in (4.17) (see Proposition 5.1). Alternatively, we could use a $\mathcal{GP}$ surrogate model and the (negative) standard deviation of the predictive distribution as the exploration function, i.e. $z_N(\boldsymbol{x}) = -\sqrt{\Sigma_{f_N}(\boldsymbol{x})}$ (see Section 2.5.2 and Section 3.4.2). The same rationale is followed by the lower confidence bound acquisition function in (2.68) for Bayesian optimization, although the terms $\hat{f}_N(\boldsymbol{x}) = \mu_{f_N}(\boldsymbol{x})$ and $z_N(\boldsymbol{x})$ are not rescaled in any way. In practice, the standard deviation of the predictive distribution is minimal at the samples in $\mathcal{X}$ (respectively, maximal for $z_N(\boldsymbol{x})$), hence selecting $\mathcal{X}_{aug} = \mathcal{X}$ does not constitute the best course of action.

- Generate $\mathcal{X}_{aug}$ by randomly sampling Ω. This rationale is followed by MSRS [116], see the acquisition function in (2.52).

- Find the global minimizer(s) and maximizer(s) of both $\hat{f}_N(\boldsymbol{x})$ and $z_N(\boldsymbol{x})$ in (5.35) and build $\mathcal{X}_{aug}$ from them. Clearly, among the proposed strategies for selecting $\mathcal{X}_{aug}$, this is the most computationally expensive, but also the most precise: the conditions in (5.36) become exact equalities. In practice, it is often not required to rescale $\hat{f}_N(\boldsymbol{x})$ and $z_N(\boldsymbol{x})$ in (5.35) so accurately, see for example the results in Section 5.2, Figure 22.

5.4.2 Proper exploration functions

The aim of $z_N(\boldsymbol{x})$ in (5.35) is to drive the optimization procedure towards those regions of Ω of the GOP (2.1) where few samples are present. To do so, the exploration function must use the information available at the current iteration, i.e. $\mathcal{X}$ in (2.9) and, possibly but not necessarily, either the measures of the cost function $\mathcal{Y}$ in (2.10) or the preferences in $\mathcal{B}$ (3.9) and $\mathcal{S}$ (3.10). We provide the following Definition to highlight which functions $z_N(\boldsymbol{x})$ are suitable to be used as exploration functions for the acquisition function in (5.35).

Definition 5.4: Proper exploration function. *Let Ω be a compact subset of $\mathbb{R}^n$ and $\mathcal{X}$ be defined as in* (2.9). *Then, a function $z_N : \mathbb{R}^n \to \mathbb{R}$ is said to be a proper exploration function if it is continuous and the solution of the pure exploration problem, i.e.:*

$$\begin{aligned} \boldsymbol{x}_{N+1} = \arg \min_{\boldsymbol{x}} & \; z_N(\boldsymbol{x}) \\ s.t. \quad & \boldsymbol{x} \in \Omega, \end{aligned} \tag{5.38}$$

is not already present in $\mathcal{X}$, that is $\boldsymbol{x}_{N+1} \notin \mathcal{X}$[a].

[a]In practice, Problem (5.38) could have multiple solutions. When that is the case, we simply require that at least one of them is not in $\mathcal{X}$ and select the latter as the new candidate sample $\boldsymbol{x}_{N+1}$.

We remark that, in the previous Definition, compactness of Ω and continuity of $z_N(\boldsymbol{x})$ are technical assumptions that ensure the existence of a solution of Problem (5.38).
Some examples of proper exploration functions are:

- The *IDW distance function* in (4.17),

$$z_N(\boldsymbol{x}) = \begin{cases} 0 & \text{if } \boldsymbol{x} \in \mathcal{X} \\ -\frac{2}{\pi} \cdot \arctan\left\{\left[\sum_{i=1}^{N} \|\boldsymbol{x} - \boldsymbol{x}_i\|_2^{-2}\right]^{-1}\right\} & \text{otherwise} \end{cases}, \tag{5.39}$$

  which is differentiable everywhere (see Proposition 4.4) and thus continuous. Furthermore, as proven by Proposition 5.1, every $\boldsymbol{x}_i \in \mathcal{X}$ is a global maximizer of (5.39). Hence, $\boldsymbol{x}_{N+1}$ obtained by Problem (5.38) is not already contained in $\mathcal{X}$.

- The *exploration function used by* `MSRS` *[116]*, in (2.51):

$$z_N(\boldsymbol{x}) = -\min_{\boldsymbol{x}_i \in \mathcal{X}} \|\boldsymbol{x} - \boldsymbol{x}_i\|_2 \, . \tag{5.40}$$

  Clearly, $z_N(\boldsymbol{x})$ in (5.40) is continuous since it is the composition of continuous functions. Furthermore, it holds that $z_N(\boldsymbol{x}_i) = 0, \forall \boldsymbol{x}_i \in \mathcal{X}$ and $z_N(\boldsymbol{x}) < 0, \forall \boldsymbol{x} \in \mathbb{R}^n \setminus \mathcal{X}$. Hence, $z_N(\boldsymbol{x})$ in (5.40) is a proper exploration function.

- Consider a surrogate model of $f(\boldsymbol{x})$ for the GOP (2.1) that is based on Gaussian processes (see Section 2.5.2 and Section 3.4.2). We can use *the (negative) standard deviation of the predictive distribution* as a proper exploration function. For example, in BBO, keeping in mind the predictive distribution of $f(\boldsymbol{x})$ in (2.44), we define:

$$\begin{aligned} z_N(\boldsymbol{x}) &= -\sqrt{\Sigma_{f_N}(\boldsymbol{x})} \\ &= -\sqrt{k_f\left(\boldsymbol{x}, \boldsymbol{x}; \boldsymbol{\theta}_f\right) - \boldsymbol{k}_f\left(\boldsymbol{x}; \boldsymbol{\theta}_f\right)^\top \cdot \left[K_f\left(\boldsymbol{\theta}_f\right) + \sigma_f^2 \cdot I_N\right]^{-1} \cdot \boldsymbol{k}_f\left(\boldsymbol{x}; \boldsymbol{\theta}_f\right)}, \end{aligned} \tag{5.41}$$

  where:

  - $k_f\left(\cdot, \cdot; \boldsymbol{\theta}_f\right)$ is the chosen kernel and $\boldsymbol{\theta}_f$ is a vector of hyper-parameters (see Theorem 2.2),
  - $\boldsymbol{k}_f\left(\cdot; \boldsymbol{\theta}_f\right)$ is the kernel vector defined in (2.45),
  - $K_f\left(\boldsymbol{\theta}_f\right)$ is the Gram matrix in (2.37),
  - $\sigma_f^2 \in \mathbb{R}_{\geq 0}$ is the variance of the Gaussian noise which affects the measures of the black-box cost function (see Assumption 2.5).

  Note that, if $k_f\left(\cdot, \cdot; \boldsymbol{\theta}_f\right)$ is continuous, then so is $z_N(\boldsymbol{x})$ in (5.41). We also give an intuitive explanation as to why the minimization of $z_N(\boldsymbol{x})$ in (5.41) leads to new candidate samples that are not already contained in $\mathcal{X}$. Roughly speaking, the $\mathcal{GP}$ predictions are the least uncertain in the neighborhoods of the observations in:

$$\mathcal{D} = \left\{(\boldsymbol{x}_i, y_i) : \boldsymbol{x}_i \in \mathcal{X}, y_i \in \mathcal{Y}, i = 1, \ldots, N\right\}.$$

  Formally, in [148], the author proves that the predictive variance at a sample $\boldsymbol{x} \in \mathbb{R}^n$, namely $\Sigma_{f_N}(\boldsymbol{x})$, cannot increase by adding points to $\mathcal{X}$, i.e.:

$$\Sigma_{f_N}(\boldsymbol{x}) \leq \Sigma_{f_{N-1}}(\boldsymbol{x}), \quad \forall N \in \mathbb{N}, N > 1.$$

  Furthermore, since $\Sigma_{f_N}(\boldsymbol{x})$ is the variance of a Gaussian distribution, we have that:

$$0 \leq \Sigma_{f_N}(\boldsymbol{x}) \leq \Sigma_{f_{N-1}}(\boldsymbol{x}), \quad \forall N \in \mathbb{N}, N > 1.$$

It follows that:

$$\Sigma_{f_N}(\boldsymbol{x}) \leq \Sigma_{f_1}(\boldsymbol{x}) = k_f\left(\boldsymbol{x}, \boldsymbol{x}; \boldsymbol{\theta}_f\right) - \frac{k_f\left(\boldsymbol{x}_1, \boldsymbol{x}; \boldsymbol{\theta}_f\right)^2}{k_f\left(\boldsymbol{x}_1, \boldsymbol{x}_1; \boldsymbol{\theta}_f\right) + \sigma_f^2}, \quad \forall N \in \mathbb{N}, N > 1.$$

Now, let us evaluate the variance at $\boldsymbol{x}_1 \in \mathcal{X}$ and suppose that the measures of $f(\boldsymbol{x})$ are not affected by noise (i.e. $\sigma_f^2 = 0$), then:

$$\Sigma_{f_N}(\boldsymbol{x}_1) \leq \Sigma_{f_1}(\boldsymbol{x}_1) = k_f\left(\boldsymbol{x}_1, \boldsymbol{x}_1; \boldsymbol{\theta}_f\right) - \frac{k_f\left(\boldsymbol{x}_1, \boldsymbol{x}_1; \boldsymbol{\theta}_f\right)^{\cancel{2}}}{\cancel{k_f\left(\boldsymbol{x}_1, \boldsymbol{x}_1; \boldsymbol{\theta}_f\right)}} = 0, \quad \forall N \in \mathbb{N}, N > 1.$$

Thus, $\Sigma_{f_N}(\boldsymbol{x}_1)$ must be zero, since it is a non-negative quantity. In conclusion, if $\sigma_f^2 = 0$ (and the Gram matrix $K_f\left(\boldsymbol{\theta}_f\right)$ in (2.37) is positive-definite), then the $\mathcal{GP}$ surrogate model interpolates the samples in $\mathcal{D}$, resulting in $\Sigma_{f_N}(\boldsymbol{x}_i) = 0, \forall \boldsymbol{x}_i \in \mathcal{X}$ and $\Sigma_{f_N}(\boldsymbol{x}) \geq 0, \forall \boldsymbol{x} \in \mathbb{R}^n \setminus \mathcal{X}$. Then, the solutions of Problem (5.38) for $z_N(\boldsymbol{x})$ in (5.41) are not already contained in $\mathcal{X}$, making it a proper exploration function. In practice, even if the measures of $f(\boldsymbol{x})$ are not affected by noise, the term $\sigma_f^2 \cdot I_N$ is often added to the Gram matrix $K_f\left(\boldsymbol{\theta}_f\right)$ to guarantee its invertibility and/or avoid ill-conditioning [153]. Hence, $\Sigma_{f_N}(\cdot)$ might not necessarily be zero at $\boldsymbol{x}_i \in \mathcal{X}$ but surely it does not assume its maximal value there.

Figure 25 compares the proper exploration functions in (5.39), (5.40) and (5.41). The global maximizer of both $z_N(\boldsymbol{x})$ in (5.39) and in (5.40) is (approximately) $x = 1.5$, whereas $z_N(\boldsymbol{x})$ in (5.41) assumes its maximal value at $x = 0.5$.

5.4.3 General scheme and convergence

The `gMRS` [108] optimization scheme is described in Algorithm 15. Due to its general nature, we have opted to keep the description of the procedure more abstract, omitting some minor algorithmic details such as the rescaling of the optimization problem and the recalibration of the surrogate model, although these can be easily included. Notice how the `GLIS-r` [108] and `GLISp-r` [109] methods, in Algorithm 14, follow the `gMRS` [108] scheme.

Convergence

The global convergence of the `gMRS` [108] scheme can be addressed in a similar fashion to the Proof of Theorem 5.2 for algorithms `GLIS-r` [108] and `GLISp-r` [109]. In what follows, we use the same notation of Section 5.3.1 and, consistently with Assumption 2.4, we suppose that, in BBO, the measures of the black-box cost function are not affected by noise. We start by proving that any algorithm that solves the pure exploration Problem (5.38) infinitely often and with a proper $z_N(\boldsymbol{x})$ generates a sequence of iterates that is dense in Ω.

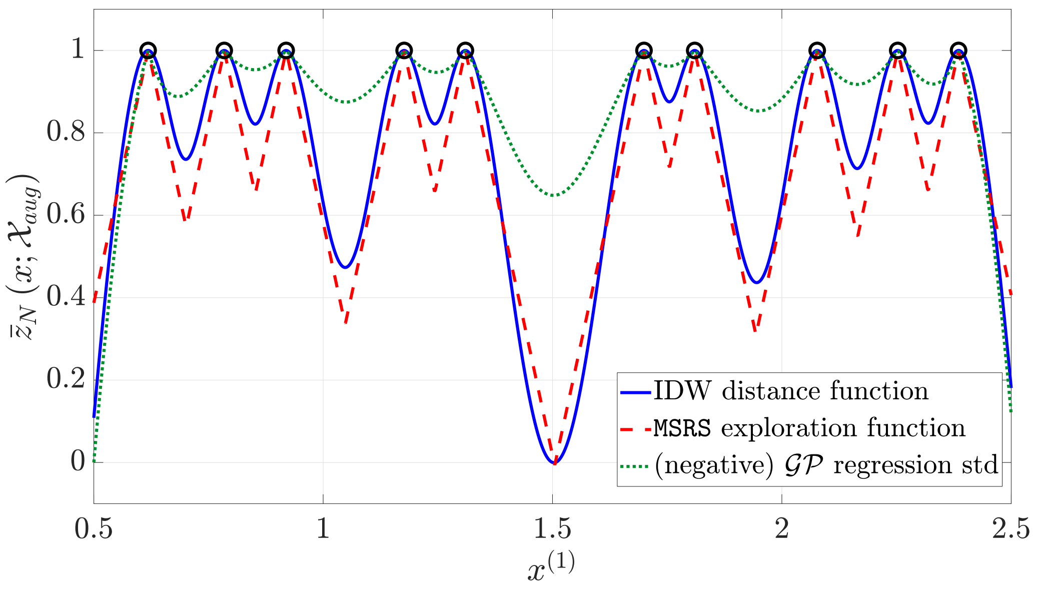


Figure 25: One-dimensional comparison between the proper exploration functions in (5.39), (5.40) **and** (5.41) **for** $N = 10$. **Concerning** $z_N(\boldsymbol{x})$ **in** (5.41)**, we have used the squared exponential kernel:** $k_f(\boldsymbol{x}, \tilde{\boldsymbol{x}}; \boldsymbol{\theta}_f) = \left(\theta_f^{(1)}\right)^2 \cdot \exp\left\{-\frac{1}{2} \cdot \left(\frac{\|\boldsymbol{x}-\tilde{\boldsymbol{x}}\|_2}{\theta_f^{(2)}}\right)^2\right\}$. **In particular, we have fixed** $\sigma_f = 10^{-6}$ **for** (5.41)**, while the hyper-parameters** $\boldsymbol{\theta}_f$ **are computed through maximum likelihood estimation (i.e. as in Problem** (2.41)**). All** $z_N(\boldsymbol{x})$**'s have been rescaled using min-max normalization (see Definition 5.1 and** $a_N(\boldsymbol{x})$ **in** (5.35)**), with** $\mathcal{X}_{aug}$ **generated by Algorithm 13 using** $K_{aug} = 5, l = 0.5, u = 2.5$.

Proposition 5.3. *Let Ω be a compact subset of $\mathbb{R}^n$ and let $\langle \boldsymbol{x}_i \rangle_{i\geq 1}$ be the sequence of iterates generated by an algorithm* `A` *(when run indefinitely). Consider a proper exploration function $z_N(\boldsymbol{x})$ (as in Definition 5.4) and suppose that there exists a strictly increasing sequence of positive integers $\langle i_t \rangle_{t\geq 1}, i_t \in \mathbb{N}$, such that the samples $\boldsymbol{x}_{i_t} \in \mathcal{X}_\infty, t \in \mathbb{N}$, are found by solving the following optimization problem (pure exploration):*

$$\begin{aligned} \boldsymbol{x}_{i_t} &= \arg\min_{\boldsymbol{x}} z_{i_t-1}(\boldsymbol{x}), \quad \forall t \in \mathbb{N} \\ s.t. \quad & \boldsymbol{x} \in \Omega. \end{aligned} \tag{5.42}$$

Then, $\mathcal{X}_\infty$ generated by `A` *is dense in Ω.*

Proof. This Proposition can be proven by applying the definition of proper exploration function and following the same rationale used in the Proof of Theorem 5.2.

From Definition 5.4, we have that Problem (5.42) admits at least a solution that is not already contained in $\mathcal{X}_{i_t-1}$. Select $\boldsymbol{x}_{i_t} \in \Omega$ as one of such solutions, then:

$$\boldsymbol{x}_{i_t} \notin \mathcal{X}_{i_t-1} \implies \exists \epsilon \in \mathbb{R}_{>0} \text{ such that } \min_{1\leq i\leq i_t-1} \left\|\boldsymbol{x}_{i_t} - \boldsymbol{x}_i\right\|_2 \geq \epsilon, \quad \forall t \in \mathbb{N}. \tag{5.43}$$

By combining (5.24) and (5.43), we get:

$$0 < \epsilon \leq d_\Omega \left(\mathcal{X}_{i_t - 1} \right), \quad \forall t \in \mathbb{N}. \tag{5.44}$$

Therefore, $\exists \alpha \in (0, 1]$ such that $\epsilon = \alpha \cdot d_\Omega \left(\mathcal{X}_{i_t - 1} \right), \forall t \in \mathbb{N}$. The condition (5.17) of Theorem 5.1 is satisfied and thus algorithm `A` produces a sequence of iterates that is dense in Ω. □

Finally, we state the convergence Theorem for the `gMRS` [108] scheme.

Theorem 5.3: Convergence of `gMRS` [108]

Let $\Omega \subset \mathbb{R}^n$ be a compact set and either:

- *$f : \mathbb{R}^n \to \mathbb{R}$ be a continuous function (BBO) or*
- *$\succsim$ be a continuous preference relation on Ω of a rational (as in Definition 3.1) human decision-maker (PBO).*

If $z_N(\boldsymbol{x})$ for the acquisition function in (5.35) *is a proper exploration function (as in Definition 5.4) and $\exists \delta_j \in \Delta_{cycle}$ in* (5.15) *such that $\delta_j = 0$, then, for $N_{max} \to \infty$, Algorithm 15 converges to the global minima of the GOP* (2.1) *for any continuous surrogate model $\hat{f}_N(\boldsymbol{x})$.*

Proof. Compactness of Ω, continuity of $f(\boldsymbol{x})$ or $\succsim$ and rationality of the DM are conditions that ensure the existence of a solution for the GOP (2.1). Furthermore, continuity of $\hat{f}_N(\boldsymbol{x})$ and $z_N(\boldsymbol{x})$ guarantee that Problem (5.37) always admits a solution.
Consider the sequence of iterates $\langle \boldsymbol{x}_i \rangle_{i \geq 1}$ produced by Algorithm 15. The first $N_{init} \in \mathbb{N}$ elements of $\langle \boldsymbol{x}_i \rangle_{i \geq 1}$ are obtained by the experimental design. Instead, each $\boldsymbol{x}_i \in \mathcal{X}_\infty, i > N_{init}$, is selected as the solution of Problem (5.37), cycling δ in (5.35) as proposed in Section 5.2.3. Following the same rationale adopted in the Proof of Theorem 5.2, we consider two strictly increasing sequences of positive integers:

- $\langle i_{t''} \rangle_{t''=1}^{t''_{max}}, t''_{max} \in \mathbb{N}$, which represents the indexes of those samples that improve upon the best candidates $\boldsymbol{x_{best}}(i_{t''} - 1), \forall t'' : 1 \leq t'' \leq t''_{max}$, see (5.32).
- $\langle i_{t'} \rangle_{t' \geq 1}$, which represents the indexes of those samples found by solving the pure exploration problem in (5.38) (i.e. for $\delta = 0$ in (5.35)).

We distinguish two cases: (i) if Algorithm 15 always improves upon its current best candidate, then $\langle i_{t''} \rangle_{t''=1}^{t''_{max}}$ is actually an infinite sequence; (ii) otherwise, due to the greedy δ-cycling in (5.16), the pure exploration problem in (5.38) is solved infinitely often. In any of the aforementioned cases, $\mathcal{X}_\infty$ is dense in Ω, see the Proof of Theorem 5.2 and Proposition 5.3. Hence, by Theorem 1.2, Algorithm 15 converges to the global minima of the GOP (2.1). $\square$

Clearly, since gMRS [108] is globally convergent, then so are GLIS-r [108] and GLISp-r [109], as proven in Section 5.3.1.

Algorithm 15: gMRS [108]

Input: (i) A-priori known constraint set Ω of the GOP (2.1); (ii) Number of initial samples $N_{init} \in \mathbb{N}$; (iii) Budget $N_{max} \in \mathbb{N}, N_{max} > N_{init}$; (iv) Continuous surrogate model $\hat{f}_N(\boldsymbol{x})$ with, possibly, its hyper-parameters; (v) Proper exploration function $z_N(\boldsymbol{x})$ (see Definition 5.4); (vi) Cycling set Δ_{cycle} in (5.15) for the acquisition function $a_N(\boldsymbol{x})$ in (5.35).

Output: (i) Best cost obtained by the procedure $y_{best}(N_{max})$ (only for BBO); (ii) Best sample obtained by the procedure $\boldsymbol{x_{best}}(N_{max})$.

```
1:  Generate a set $\mathcal{X}$ in (2.9) of $N_{init}$ starting points using a suitable experimental design (see Section 2.4)
2:  Evaluate the samples in $\mathcal{X}$ either by measuring the values of $f(\cdot)$, obtaining the set $\mathcal{Y}$ in (2.10) (BBO), or
    by querying the decision-maker as in Algorithm 7, obtaining the sets $\mathcal{B}$ (3.9) and $\mathcal{S}$ (3.10), as well as the
    best candidate $\boldsymbol{x_{best}}(N_{init})$
3:  Set $N = N_{init}$ (and $M = |\mathcal{B}|$ for PBO)
4:  Set $\delta = \delta_0 \in \Delta_{cycle}$ and $j = 0$
5:  for $k = 1, 2, \dots, N_{max} - N_{init}$ do
6:      Build the surrogate model $\hat{f}_N(\boldsymbol{x})$ from $\mathcal{X}$ and the information on $f(\boldsymbol{x})$ at hand
7:      Generate the augmented sample set $\mathcal{X}_{aug}$
8:      Look for the next candidate sample $\boldsymbol{x}_{N+1}$ by solving Problem (5.37) with $a_N(\boldsymbol{x})$ in (5.35)
9:      Evaluate the new candidate sample, obtaining either $y_{N+1} = f(\boldsymbol{x}_{N+1})$ (in BBO) or $b_{M+1} =$
        $\pi_{\succsim}(\boldsymbol{x}_{N+1}, \boldsymbol{x_{best}}(N))$ (in PBO)
10:     Update the set of samples $\mathcal{X}$ and either the set of measures of $f(\cdot)$, $\mathcal{Y}$ (BBO), or the preference
        information in the sets $\mathcal{B}$ and $\mathcal{S}$ (PBO)
11:     Check if $\boldsymbol{x}_{N+1}$ improves upon $\boldsymbol{x_{best}}(N)$, as highlighted by the flag $hasImproved$ from Algorithm 9
12:     if $hasImproved$ then
13:         Set $\boldsymbol{x_{best}}(N+1) = \boldsymbol{x}_{N+1}$
14:     else
15:         Keep $\boldsymbol{x_{best}}(N+1) = \boldsymbol{x_{best}}(N)$
16:         Set $\delta = \delta_{(j+1) \bmod N_{cycle}} \in \Delta_{cycle}$ (greedy $\delta$-cycling) and $j = j + 1$
17:     Set $N = N + 1$ (and $M = M + 1$ for PBO)
```

5.5 Chapter summary

In this Chapter, we have described the second and third contributions of this book, namely the extension of the GLIS [10] and GLISp [11] procedures, resulting in the GLIS-r [108] and GLISp-r [109] algorithms, and the derivation of a general surrogate-based scheme for unconstrained black-box and preference-based optimization, which we have called gMRS [108]. The main novelty in this dissertation is the assessment of the global convergence of the proposed PBO methods (and not just the BBO ones). Notably, to the best of our knowledge, no proof of convergence is available for any of the preference-based response surface techniques described in Chapter 3. Here, we have seen how, by leveraging some results taken from the utility theory framework (Section 3.1 and Section 3.2), it is possible to address the convergence of a PBO procedure as we would for any global optimization method.

For what concerns `GLIS` [10] and `GLISp` [11] specifically, we have identified two shortcomings of the IDW distance function in (4.17). In `GLIS-r` [108] and `GLISp-r` [109], we have addressed the limitations of $z_N(\boldsymbol{x})$ in (4.17) through the definition of a revisited infill sampling criterion. Moreover, we have also proposed a novel way of alternating between exploration and exploitation, namely the greedy δ-cycling approach, which is aimed at making `GLIS-r` [108] and `GLISp-r` [109] more sample efficient. The proposed extensions will be compared to `GLIS` [10] and `GLISp` [11] on several benchmark optimization problems in Chapter 7. Therein, we will show that the modifications suggested in this Chapter can make `GLIS-r` [108] and `GLISp-r` [109] more robust than the original methods, especially in the preference-based setting.

Chapter 6. Handling black-box constraints in GLIS-r and GLISp-r

This Chapter is devoted to describing the fourth contribution of this book, which is the *extension of the GLIS-r [108] and GLISp-r [109] procedures (see Chapter 5) to the constrained black-box and preference-based optimization frameworks*. We refer to the proposed extensions as C-GLIS-r and C-GLISp-r respectively. Similarly to Chapter 5, we treat BBO and PBO in a unified fashion.

In what follows, we assume that the black-box constraints functions of the GOP (2.1) are not measurable, i.e. the set $\mathcal{C}_\Xi$ in (2.12) is not available. Instead, we only have at our disposal the set $\mathcal{U}_\Xi$ in (2.11), which discerns the Ξ-feasible samples in $\mathcal{X}$ (2.9) from the Ξ-infeasible ones. The rationale behind this choice is that, in PBO, if only decision-maker-based constraints are present (see Definition 3.6), then $\mathcal{C}_\Xi$ and $\mathcal{U}_\Xi$ carry the same information. We build a surrogate model for the black-box constraints functions in a similar fashion to the algorithm proposed in [9]: $\mathcal{X}$ and $\mathcal{U}_\Xi$ are used to train a Probabilistic Support Vector Machine (PSVM) classifier [15, 106], which estimates the probability of Ξ-feasibility, $p_N(\boldsymbol{x} \in \Xi) = p\left(\boldsymbol{x} \in \Xi \middle| \mathcal{U}_\Xi, \mathcal{X}, \boldsymbol{x}\right)$, of new candidate samples. Support Vector Machines (SVMs) [15] have also been employed in [3] to approximate the feasible region of the GOP (2.1). Instead, C-GLIS and C-GLISp [156] estimate the probability of a sample being Ξ-feasible by means of IDW interpolation (see Section 4.1.3). Here, *we propose a slight modification to the probabilistic support vector machine classifier that makes it better suited for black-box and preference-based optimization.*

The infill sampling criterion of C-GLIS-r and C-GLISp-r combines ideas taken from multiple BBO algorithms (in Section 2.6):

1. We use different strategies based on whether a Ξ-feasible sample is available or not (cf. the two-phase approach of COBRA [112]);

2. We look for new candidate samples as in SuperEGO [125] and Constrained EGO with Support Vector Machines [9], i.e. by optimizing some acquisition function subject to the constraint $p_N(\boldsymbol{x} \in \Xi) \geq \gamma$, where $\gamma \in [0, 1]$ is a user-defined threshold on the estimated probability of Ξ-feasibility, see Problem (2.72) and Problem (2.74).

3. We satisfy the constraint $p_N(\boldsymbol{x} \in \Xi) \geq \gamma$ with some slack, since $p_N(\boldsymbol{x} \in \Xi) \geq \gamma$ is only an approximation of the probability of Ξ-feasibility. This rationale is different from that of COBRA [112], which looks for new candidate samples by solving Problem (2.63), wherein a margin is

added to each black-box constraint to make it less likely to find a Ξ-infeasible point, achieving a more conservative procedure.

4. We modify the greedy δ-cycling strategy of `GLIS-r` [108] and `GLISp-r` [109] to take into account that the improvement also depends on the Ξ-feasibility, see Algorithm 9.

The remainder of this Chapter is organized as follows. Section 6.1 briefly reviews how PSVMs work and presents a novel way of making these classifiers more suited for constrained black-box and preference-based optimization. We also discuss how to calibrate their hyper-parameters. Section 6.2 describes the infill sampling criterion of `C-GLIS-r` and `C-GLISp-r`. Then, Section 6.3 reports the pseudocode for the proposed methods and gives some remarks on their global convergence. Lastly, Section 6.4 summarizes the results presented in this Chapter.

6.1 Probabilistic support vector machines revisited

Before introducing the proposed revisitation of the PSVM classifier, which is tailored for constrained black-box and preference-based optimization, it is best to review some basic concepts. In Section 6.1.1, we discuss how Support Vector Machines (SVMs) [15] work. Off-the-shelf, SVMs are binary classifiers that do not return the probability of a point belonging either to one or the other class. In [106], the author gives them a probabilistic interpretation, obtaining the so-called Probabilistic Support Vector Machine (PSVM) classifier, which we review in Section 6.1.2. Finally, in Section 6.1.3, we describe the proposed revisitation of the PSVM classifier. We conclude our dissertation on PSVMs by discussing the calibration of their hyper-parameters, in Section 6.1.4.

6.1.1 Support vector machines

SVMs assume that the samples $\boldsymbol{x}_i \in \mathcal{X}$ in (2.9) either belong to a positive or a negative class, denoted by the labels $t_i = 1$ and $t_i = -1$ respectively. Therefore, instead of using the set $\mathcal{U}_\Xi$ in (2.11), we define the set of labels $\mathcal{T}_\Xi$ as:

$$\mathcal{T}_\Xi = \{t_i = \mathbb{I}(u_i = 1) - \mathbb{I}(u_i = 0), u_i \in \mathcal{U}_\Xi, i = 1, \dots, N\} \tag{6.1}$$

and the corresponding vector of labels as $\boldsymbol{t} = \begin{bmatrix} t_1 & \dots & t_N \end{bmatrix}^\top \in \{-1, 1\}^N$.

In the context of constrained BBO and PBO, support vector machines can be used to estimate the boundary of the set Ξ of the GOP (2.1). From a machine learning perspective, this boundary is referred to as the decision boundary and it separates the points belonging to the positive class (Ξ-feasible) from those of the negative class (Ξ-infeasible). In order to describe even highly nonlinear decision boundaries, SVM classifiers are equipped with kernels (see Theorem 2.2). We restrict our analysis to those kernels that rely on radial basis functions. Note that not all RBFs in Definition 2.6 can be used for that purpose, we must consider only the ones that satisfy Mercer's Theorem 2.2. For example, the Gaussian RBF is a popular choice. We consider a radial basis function expansion model $m_{\Xi_N} : \mathbb{R}^n \to \mathbb{R}$ which, differently from (2.20), also includes the *intercept* $\beta_{\Xi}^{(0)} \in \mathbb{R}$:

$$\begin{aligned} m_{\Xi_N}\left(\boldsymbol{x}; \tilde{\boldsymbol{\beta}}_{\Xi}, \epsilon_{\Xi}\right) &= \beta_{\Xi}^{(0)} + \sum_{i=1}^{N} \beta_{\Xi}^{(i)} \cdot \varphi_{\Xi}\left(\epsilon_{\Xi} \cdot \|\boldsymbol{x} - \boldsymbol{x}_i\|_2\right) \\ &= \beta_{\Xi}^{(0)} + \sum_{i=1}^{N} \beta_{\Xi}^{(i)} \cdot \phi_{\Xi_i}\left(\boldsymbol{x}; \epsilon_{\Xi}\right) \\ &= \tilde{\boldsymbol{\phi}}_{\Xi}\left(\boldsymbol{x}; \epsilon_{\Xi}\right)^{\top} \cdot \tilde{\boldsymbol{\beta}}_{\Xi}. \end{aligned} \tag{6.2}$$

In (6.2), with a slight abuse of notation, we have defined $\tilde{\boldsymbol{\beta}}_{\Xi} = \begin{bmatrix} \beta_{\Xi}^{(0)} & \boldsymbol{\beta}_{\Xi} \end{bmatrix}^{\top} \in \mathbb{R}^{N+1}$ and $\tilde{\boldsymbol{\phi}}_{\Xi}\left(\boldsymbol{x}; \epsilon_{\Xi}\right) = \begin{bmatrix} 1 & \boldsymbol{\phi}_{\Xi}\left(\boldsymbol{x}; \epsilon_{\Xi}\right) \end{bmatrix}^{\top} \in \mathbb{R}^{N+1}$. As always, $\varphi_{\Xi} : \mathbb{R}_{\geq 0} \to \mathbb{R}$ denotes the chosen radial function and $\epsilon_{\Xi} \in \mathbb{R}_{>0}$ is the shape parameter (see Definition 2.4 and Definition 2.5). The decision boundary of the SVM classifier is described by the following equation:

$$m_{\Xi_N}\left(\boldsymbol{x}; \tilde{\boldsymbol{\beta}}_{\Xi}, \epsilon_{\Xi}\right) = 0. \tag{6.3}$$

Furthermore, we can predict the class of a point $\tilde{\boldsymbol{x}} \in \mathbb{R}^n$ as:

$$\hat{\tilde{t}} = \operatorname{sign}\left\{m_{\Xi_N}\left(\tilde{\boldsymbol{x}}; \tilde{\boldsymbol{\beta}}_{\Xi}, \epsilon_{\Xi}\right)\right\}. \tag{6.4}$$

Clearly, different choices of the vector of weights $\tilde{\boldsymbol{\beta}}_{\Xi}$ in (6.2) result in different decision boundaries (which might not even correctly separate the two classes). In practice, support vector machine classifiers compute $\tilde{\boldsymbol{\beta}}_{\Xi}$ so that the margin, i.e. the smallest distance between the decision boundary and any of the samples in $\mathcal{X}$ (2.9), is maximized. For this reason, SVMs are often referred to as *maximum margin classifiers*. *Hard margin support vector machines* compute the weights $\tilde{\boldsymbol{\beta}}_{\Xi}$ in (6.2) so that all the training samples are classified correctly, i.e.

$$\operatorname{sign}\left\{m_{\Xi_N}\left(\boldsymbol{x}_i; \tilde{\boldsymbol{\beta}}_{\Xi}, \epsilon_{\Xi}\right)\right\} = t_i, \quad \forall t_i \in \mathcal{T}_{\Xi}, \boldsymbol{x}_i \in \mathcal{X},$$

or, equivalently,

$$t_i \cdot m_{\Xi_N}\left(\boldsymbol{x}_i; \tilde{\boldsymbol{\beta}}_{\Xi}, \epsilon_{\Xi}\right) \geq 1, \quad \forall t_i \in \mathcal{T}_{\Xi}, \boldsymbol{x}_i \in \mathcal{X}. \tag{6.5}$$

In practice, there might not exist a choice of $\tilde{\boldsymbol{\beta}}_{\Xi}$ in (6.2) for which (6.5) holds. *Soft margin support vector machines* compensate for this shortcoming by allowing some training samples to be misclassified. To do so, we introduce one slack variable $\varepsilon_{SVM}^{(i)} \in \mathbb{R}_{\geq 0}, i = 1, \dots, N$, for each constraint in (6.5), obtaining:

$$t_i \cdot m_{\Xi_N}\left(\boldsymbol{x}_i; \tilde{\boldsymbol{\beta}}_{\Xi}, \epsilon_{\Xi}\right) \geq 1 - \varepsilon_{SVM}^{(i)}, \quad \forall t_i \in \mathcal{T}_{\Xi}, \boldsymbol{x}_i \in \mathcal{X}. \tag{6.6}$$

In this case, we are interested in a classifier that maximizes the margin but, at the same time, penalizes the misclassification of the data in:

$$\mathcal{D} = \{(\boldsymbol{x}_i, t_i) : \boldsymbol{x}_i \in \mathcal{X}, t_i \in \mathcal{T}_\Xi, i = 1, \dots, N\}. \tag{6.7}$$

Soft margin SVMs boil down to solving the following optimization problem:

$$\begin{aligned} &\arg \min_{\boldsymbol{\alpha}_\Xi} J_{SVM}(\boldsymbol{\alpha}_\Xi) \\ \text{s.t.} \quad &\mathbf{0}_N \leq \boldsymbol{\alpha}_\Xi \leq C_{SVM} \cdot \mathbf{1}_N \\ &\boldsymbol{t}^\top \cdot \boldsymbol{\alpha}_\Xi = 0. \end{aligned} \tag{6.8}$$

The objective function $J_{SVM} : \mathbb{R}^N \to \mathbb{R}$ is:

$$\begin{aligned} J_{SVM}(\boldsymbol{\alpha}_\Xi) &= \frac{1}{2} \cdot \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_\Xi^{(i)} \cdot \alpha_\Xi^{(j)} \cdot t_i \cdot t_j \cdot \varphi_\Xi\left(\epsilon_\Xi \cdot \left\|\boldsymbol{x}_i - \boldsymbol{x}_j\right\|_2\right) - \sum_{i=1}^{N} \alpha_\Xi^{(i)} \\ &= \frac{1}{2} \cdot \boldsymbol{\alpha}_\Xi^\top \cdot \left[\left(\boldsymbol{t} \cdot \boldsymbol{t}^\top\right) \odot \Phi_\Xi(\epsilon_\Xi)\right] \cdot \boldsymbol{\alpha}_\Xi - \mathbf{1}_N^\top \cdot \boldsymbol{\alpha}_\Xi, \end{aligned} \tag{6.9}$$

where $\odot$ denotes the Hadamard product operator and $\Phi_\Xi(\epsilon_\Xi)$ is defined as in (2.26). In Problem (6.8), $\boldsymbol{\alpha}_\Xi \in \mathbb{R}^N$ are the Lagrange multipliers (one for each point $\boldsymbol{x}_i \in \mathcal{X}$ in (2.9)) associated to the constraints in (6.6). The weight $C_{SVM} \in \mathbb{R}_{>0}$ penalizes the misclassification of the training samples. In particular, low values of C_{SVM} may lead to a lower accuracy of the classifier on $\mathcal{D}$ in (6.7) but better performances on out-of-sample data. Vice-versa, for $C_{SVM} \to \infty$, we go back to the hard margin formulation of SVMs. The derivation of Problem (6.8) is quite long and out of scope of this book, the interested reader is referred to [15, 29]. Here, we simply point out the following Proposition.

Proposition 6.1. *Problem* (6.8) *is a convex QP.*

Proof. Clearly, Problem (6.8) is a quadratic program since the objective function in (6.9) is a quadratic function of $\boldsymbol{\alpha}_\Xi$ while all the constraints functions are linear in $\boldsymbol{\alpha}_\Xi$. Concerning its convexity, we need to prove that the matrix $[(\boldsymbol{t} \cdot \boldsymbol{t}^\top) \odot \Phi_\Xi(\epsilon_\Xi)]$ in (6.9) is positive semidefinite [18]. Note that $(\boldsymbol{t} \cdot \boldsymbol{t}^\top)$ is positive semidefinite by construction while $\Phi_\Xi(\epsilon_\Xi)$ is so under the assumption that $\varphi_\Xi(\cdot)$ is a proper kernel (see Mercer's Theorem 2.2). Then, by Schur Product Theorem [129], the matrix $[(\boldsymbol{t} \cdot \boldsymbol{t}^\top) \odot \Phi_\Xi(\epsilon_\Xi)]$ is also positive semidefinite. □

The weights $\beta_\Xi^{(i)}, i = 1, \dots, N$, in (6.2) can be computed directly from the Lagrange multipliers $\boldsymbol{\alpha}_\Xi$ obtained from solving Problem (6.8):

$$\beta_\Xi^{(i)} = t_i \cdot \alpha_\Xi^{(i)}, \quad i = 1, \dots, N. \tag{6.10}$$

Note that, due to the KKT conditions (see Theorem A.11), some of the constraints in (6.6) hold with equality (active constraints), in which case $\alpha_{\Xi}^{(i)} > 0$, while others hold with inequality and thus their corresponding Lagrange multipliers are zero, i.e. $\alpha_{\Xi}^{(i)} = 0$. In particular, we have that:

- If $\alpha_{\Xi}^{(i)} = 0$, then the sample $\boldsymbol{x}_i \in \mathcal{X}$ does not contribute to the predictive model in (6.2) (see (6.10));
- The remaining points $\boldsymbol{x}_i \in \mathcal{X}$, for which $0 < \alpha_{\Xi}^{(i)} \leq C_{SVM}$, constitute the <u>support vectors</u>. We highlight them through the following set of indexes:

$$\mathcal{I}_{sv} = \left\{ i : 0 < \alpha_{\Xi}^{(i)} \leq C_{SVM} \right\}, \quad |\mathcal{I}_{sv}| = N_{sv}, N_{sv} \in \mathbb{N}. \tag{6.11}$$

In particular, we remark that those support vectors indexed by $i'_{sv} \in \mathcal{I}'_{sv}$,

$$\mathcal{I}'_{sv} = \left\{ i : 0 < \alpha_{\Xi}^{(i)} < C_{SVM} \right\}, \quad \mathcal{I}'_{sv} \subseteq \mathcal{I}_{sv}, \tag{6.12}$$

are such that the slacks in (6.6) are zero, resulting in $t_{i'_{sv}} \cdot m_{\Xi_N} \left(\boldsymbol{x}_{i'_{sv}}; \tilde{\boldsymbol{\beta}}_{\Xi}, \epsilon_{\Xi} \right) = 1, \forall i'_{sv} \in \mathcal{I}'_{sv}$. Therefore, the distance between the points $\boldsymbol{x}_{i'_{sv}} \in \mathcal{X}, i'_{sv} \in \mathcal{I}'_{sv}$, and the decision boundary of the classifier is exactly the margin. Lastly, those samples $\boldsymbol{x}_i \in \mathcal{X}$ for which $\alpha_{\Xi}^{(i)} = C_{SVM}$ are associated to positive slacks in (6.6), $\epsilon_{SVM}^{(i)} \in \mathbb{R}_{>0}$, and can either be correctly classified, if $\epsilon_{SVM}^{(i)} \leq 1$, or misclassified, if $\epsilon_{SVM}^{(i)} > 1$. See Figure 26 for an example.

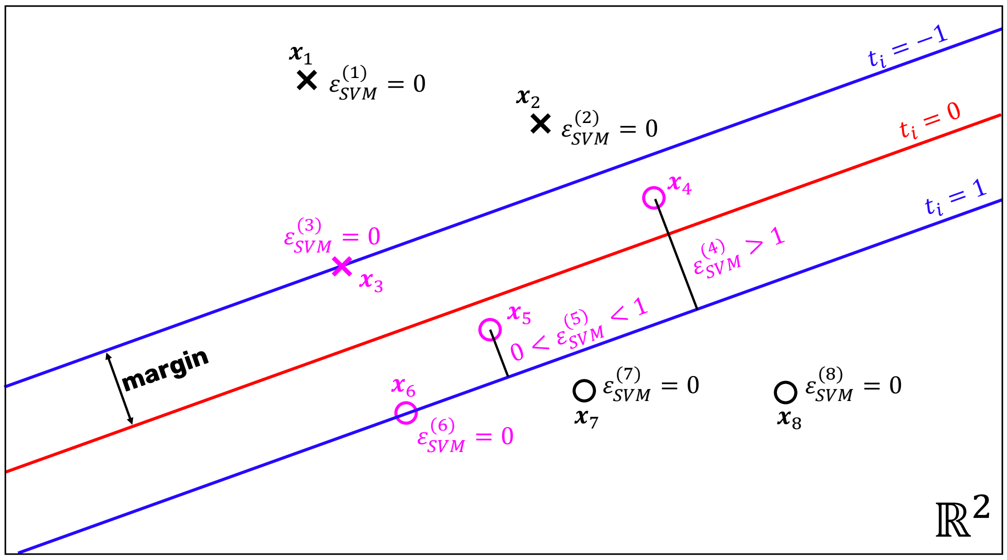

$\boldsymbol{x}_1$
$\varepsilon_{SVM}^{(1)} = 0$
$\boldsymbol{x}_2$
$\varepsilon_{SVM}^{(2)} = 0$
$t_i = -1$
$t_i = 0$
$t_i = 1$
$\boldsymbol{x}_4$
$\varepsilon_{SVM}^{(4)} > 1$
$\varepsilon_{SVM}^{(3)} = 0$
$\boldsymbol{x}_3$
$\boldsymbol{x}_5$
$0 < \varepsilon_{SVM}^{(5)} < 1$
margin
$\boldsymbol{x}_6$
$\varepsilon_{SVM}^{(6)} = 0$
$\varepsilon_{SVM}^{(7)} = 0$
$\boldsymbol{x}_7$
$\varepsilon_{SVM}^{(8)} = 0$
$\boldsymbol{x}_8$
$\mathbb{R}^2$

Figure 26: Two-dimensional example of a SVM classifier for $N = 8$ training samples. The red line denotes the decision boundary, the margin is highlighted with a black double arrow. The points belonging to the positive class are depicted with circles whereas the ones of the negative class are the crosses. The support vectors (in magenta) are the points x_3, x_4, x_5 and x_6. For each sample $x_i, i = 1, \dots, 8$, we report its slack $\varepsilon_{SVM}^{(i)}$. Note that the point x_4 is misclassified.

One of the main advantages of SVMs is that, once $\boldsymbol{\beta}_\Xi$ for (6.2) has been computed by solving Problem (6.8), the points that are not support vectors can be "discarded" from the predictive model $m_{\Xi_N}\left(\boldsymbol{x};\tilde{\boldsymbol{\beta}}_\Xi,\epsilon_\Xi\right)$ since $\beta_\Xi^{(i)}=0,\forall i\in\{1,\dots,N\}\setminus\mathcal{I}_{sv}$. Thus, we can consider:

$$m_{\Xi_N}\left(\boldsymbol{x};\tilde{\boldsymbol{\beta}}_\Xi,\epsilon_\Xi\right)=\beta_\Xi^{(0)}+\sum_{i_{sv}\in\mathcal{I}_{sv}}\beta_\Xi^{(i_{sv})}\cdot\varphi_\Xi\left(\epsilon_\Xi\cdot\left\|\boldsymbol{x}-\boldsymbol{x}_{i_{sv}}\right\|_2\right)\tag{6.13}$$

instead of (6.2). Lastly, the intercept $\beta_\Xi^{(0)}$ can be computed as [15]:

$$\beta_\Xi^{(0)}=\frac{1}{|\mathcal{I}'_{sv}|}\cdot\sum_{i'_{sv}\in\mathcal{I}'_{sv}}\left[t_{i'_{sv}}-\sum_{j_{sv}\in\mathcal{I}_{sv}}\alpha_\Xi^{(j_{sv})}\cdot t_{j_{sv}}\cdot\varphi_\Xi\left(\epsilon_\Xi\cdot\left\|\boldsymbol{x}_{i'_{sv}}-\boldsymbol{x}_{j_{sv}}\right\|_2\right)\right].\tag{6.14}$$

Now, suppose that we want to classify a point $\tilde{\boldsymbol{x}}\in\mathbb{R}^n$. SVMs produce an "uncalibrated" value $m_{\Xi_N}\left(\tilde{\boldsymbol{x}};\tilde{\boldsymbol{\beta}}_\Xi,\epsilon_\Xi\right)$ and predict the label of $\tilde{\boldsymbol{x}}$ as in (6.4).

6.1.2 Probabilistic support vector machines

We are interested in converting the "uncalibrated" values returned by SVMs to the probabilities of belonging to either the negative or the positive class. One way to do so is to employ Platt scaling [106], giving rise to probabilistic support vector machines, which operate as follows. First of all, we still train a SVM classifier as described in Section 6.1.1. Then, we fit a *sigmoid model*, $s_{\Xi_N}:\mathbb{R}^n\to[0,1]$, on the outputs of $m_{\Xi_N}\left(\boldsymbol{x};\tilde{\boldsymbol{\beta}}_\Xi,\epsilon_\Xi\right)$ in (6.13) using the data in $\mathcal{X}$ (2.9) and $\mathcal{U}_\Xi$ (2.11) (instead of the targets $\mathcal{T}_\Xi$ in (6.1)). The result is a function that estimates the probability of a sample $\boldsymbol{x}\in\mathbb{R}^n$ belonging to the positive class (Ξ-feasibility):

$$\begin{aligned}p_N(\boldsymbol{x}\in\Xi)&=p\left(\boldsymbol{x}\in\Xi\,\middle|\,\mathcal{U}_\Xi,\mathcal{X},\boldsymbol{x}\right)\\&=s_{\Xi_N}\left(\boldsymbol{x};\boldsymbol{\chi}_\Xi,\epsilon_\Xi\right)\\&=\frac{1}{1+\exp\left\{\chi_\Xi^{(1)}\cdot m_{\Xi_N}\left(\boldsymbol{x};\tilde{\boldsymbol{\beta}}_\Xi,\epsilon_\Xi\right)+\chi_\Xi^{(2)}\right\}}.\end{aligned}\tag{6.15}$$

$\boldsymbol{\chi}_\Xi=\begin{bmatrix}\chi_\Xi^{(1)} & \chi_\Xi^{(2)}\end{bmatrix}^\top\in\mathbb{R}^2$ in (6.15) is a vector of weights that is found through *maximum likelihood estimation*. Similarly to logistic regression [58], we define the negative log-likelihood $J_{PSVM}:\mathbb{R}^2\to\mathbb{R}$ as:

$$J_{PSVM}\left(\boldsymbol{\chi}_\Xi\right)=-\sum_{i=1}^N\left[u_i\cdot\ln s_{\Xi_N}\left(\boldsymbol{x}_i;\boldsymbol{\chi}_\Xi,\epsilon_\Xi\right)+(1-u_i)\cdot\ln\left(1-s_{\Xi_N}\left(\boldsymbol{x}_i;\boldsymbol{\chi}_\Xi,\epsilon_\Xi\right)\right)\right].\tag{6.16}$$

Then, we compute $\boldsymbol{\chi}_\Xi$ in (6.15) by solving the following unconstrained optimization problem:

$$\begin{aligned}&\arg\min_{\boldsymbol{\chi}_\Xi} J_{PSVM}\left(\boldsymbol{\chi}_\Xi\right)\\&\text{s.t.}\quad\boldsymbol{\chi}_\Xi\in\mathbb{R}^2.\end{aligned}\tag{6.17}$$

Note that we can use derivative-based optimization procedures to solve Problem (6.17), since $J_{PSVM}\left(\chi_\Xi\right)$ in (6.16) is twice differentiable everywhere. In particular, its gradient and Hessian are [83]:

$$\nabla_{\chi_\Xi} J_{PSVM}\left(\chi_\Xi\right) = \sum_{i=1}^{N} \begin{bmatrix} m_{\Xi_N}\left(\boldsymbol{x}_i; \tilde{\boldsymbol{\beta}}_\Xi, \epsilon_\Xi\right) \\ 1 \end{bmatrix} \cdot \left\{u_i - s_{\Xi_N}\left(\boldsymbol{x}_i; \chi_\Xi, \epsilon_\Xi\right)\right\}, \tag{6.18a}$$

$$\nabla^2_{\chi_\Xi \chi_\Xi} J_{PSVM}\left(\chi_\Xi\right) = \sum_{i=1}^{N} \begin{bmatrix} m_{\Xi_N}\left(\boldsymbol{x}_i; \text{—}\right)^2 & m_{\Xi_N}\left(\boldsymbol{x}_i; \text{—}\right) \\ m_{\Xi_N}\left(\boldsymbol{x}_i; \text{—}\right) & 1 \end{bmatrix} \cdot s_{\Xi_N}\left(\boldsymbol{x}_i; \text{—}\right) \cdot \left[1 - s_{\Xi_N}\left(\boldsymbol{x}_i; \text{—}\right)\right]. \tag{6.18b}$$

Although Problem (6.17) is nonlinear in χ_Ξ, it is possible to prove its convexity, as highlighted by the following Proposition.

Proposition 6.2. *The Hessian* $\nabla^2_{\chi_\Xi \chi_\Xi} J_{PSVM}\left(\chi_\Xi\right)$ *in* (6.18b) *is positive semidefinite. In addition,* $\nabla^2_{\chi_\Xi \chi_\Xi} J_{PSVM}\left(\chi_\Xi\right)$ *is positive definite if and only if:*

$$\min_{\boldsymbol{x}_i \in \mathcal{X}} m_{\Xi_N}\left(\boldsymbol{x}_i; \tilde{\boldsymbol{\beta}}_\Xi, \epsilon_\Xi\right) \neq \max_{\boldsymbol{x}_i \in \mathcal{X}} m_{\Xi_N}\left(\boldsymbol{x}_i; \tilde{\boldsymbol{\beta}}_\Xi, \epsilon_\Xi\right).$$

Therefore, Problem (6.17) *is convex.*

Proof. See [83]. □

We can predict the membership of a sample $\tilde{\boldsymbol{x}} \in \mathbb{R}^n$ to either the positive or the negative class by computing the probability in (6.15). In particular, we choose a threshold $\gamma \in [0, 1]$ (typically $\gamma = 0.5$) and define the surrogate Ξ-feasibility function $\hat{u}_{\Xi_N} : \mathbb{R}^n \to \{0, 1\}$ as follows:

$$\hat{u}_{\Xi_N}\left(\boldsymbol{x}\right) = \begin{cases} 1 & \text{if } p_N\left(\boldsymbol{x} \in \Xi\right) \geq \gamma \\ 0 & \text{if } p_N\left(\boldsymbol{x} \in \Xi\right) < \gamma \end{cases}. \tag{6.19}$$

In practice, $\hat{u}_{\Xi_N}\left(\boldsymbol{x}\right)$ in (6.19) is an approximation of the Ξ-feasibility function $u_\Xi\left(\boldsymbol{x}\right)$ in (2.4). Moreover, note that the decision boundary of the PSVM classifier is described by the equation:

$$p_N\left(\boldsymbol{x} \in \Xi\right) = \gamma \tag{6.20}$$

instead of (6.3), which is related to the SVM classifier.

The next Remark gives some insights on how to improve the quality of the predictions returned by the PSVM classifier on out-of-sample data.

Remark 6.1. *When* χ_Ξ *is estimated by solving Problem* (6.17)*, the resulting probability distribution* $p_N\left(\boldsymbol{x} \in \Xi\right)$ *in* (6.15) *might be biased to the training data:*

$$\mathcal{D} = \left\{\left(\boldsymbol{x}_i, u_i\right) : \boldsymbol{x}_i \in \mathcal{X}, u_i \in \mathcal{U}_\Xi, i = 1, \ldots, N\right\}, \tag{6.21}$$

leading to possibly "bad" out-of-sample predictions. That is due to the fact that the sigmoid $s_{\Xi_N}\left(\boldsymbol{x}; \boldsymbol{\chi}_{\Xi}, \epsilon_{\Xi}\right)$ *in* (6.15) *is fitted from the outputs of the model* $m_{\Xi_N}\left(\boldsymbol{x}; \tilde{\boldsymbol{\beta}}_{\Xi}, \epsilon_{\Xi}\right)$ *in* (6.13) *which, in turn, has already been trained on the same data. In [106], the author proposes two strategies to mitigate this issue:*

1. *Fit the sigmoid* $s_{\Xi_N}\left(\boldsymbol{x}; \boldsymbol{\chi}_{\Xi}, \epsilon_{\Xi}\right)$ *in* (6.15) *on a dataset that is different from the one used to train the underlying SVM. One way to generate such dataset is through cross-validation [58];*

2. *In the case of unbalanced datasets, instead of using* $u_i \in \{0, 1\}$ *for the cost function* $J_{PSVM}\left(\boldsymbol{\chi}_{\Xi}\right)$ *in* (6.16), *we can fit the sigmoid from the labels:*

$$\tilde{u}_i = \begin{cases} \frac{N_+ + 1}{N_+ + 2} & \text{if } u_i = 1 \\ \frac{1}{N_- + 2} & \text{if } u_i = 0 \end{cases},$$

where $N_+, N_- \in \mathbb{N}$ *are, respectively, the number of samples belonging to the positive and the negative class.*

In practice, when using PSVMs to estimate the probability of Ξ*-feasibility of a sample for BBO and PBO, few observations are available to train the classifier. Moreover, in* `C-GLIS-r` *and* `C-GLISp-r`*, we take into account that* $p_N\left(\boldsymbol{x} \in \Xi\right)$ *in* (6.15) *is only an approximation of the real probability of* Ξ*-feasibility and define an infill sampling criterion accordingly (see Section 6.2). For these reasons, we are not particularly concerned about the (possibly) biased estimates returned by PSVMs and we have opted to keep computing* $\boldsymbol{\chi}_{\Xi}$ *by simply solving Problem* (6.17), *using the data in* $\mathcal{D}$ (6.21). *This also avoids potential computational overheads due to performing cross-validation at each iteration of the BBO and PBO procedures.*

We conclude this Section by pointing out the following Proposition and Lemma, which deal with the differentiability of $s_{\Xi_N}\left(\boldsymbol{x}; \boldsymbol{\chi}_{\Xi}, \epsilon_{\Xi}\right)$ in (6.15) with respect to $\boldsymbol{x}$. The latter becomes relevant when dealing with the optimization problem associated to the infill sampling criterion of the `C-GLIS-r` and `C-GLISp-r` algorithms (in Section 6.2).

***Proposition 6.3: Differentiability of the sigmoid in* (6.15).** *The sigmoid* $s_{\Xi_N}\left(\boldsymbol{x}; \boldsymbol{\chi}_{\Xi}, \epsilon_{\Xi}\right)$ *in* (6.15) *is differentiable everywhere with respect to* $\boldsymbol{x}$ *if the chosen radial basis function* $\phi_{\Xi_i}\left(\boldsymbol{x}; \epsilon_{\Xi}\right) = \varphi_{\Xi}\left(\epsilon_{\Xi} \cdot \|\boldsymbol{x} - \boldsymbol{x}_i\|_2\right)$ *for the model* $m_{\Xi_N}\left(\boldsymbol{x}; \tilde{\boldsymbol{\beta}}_{\Xi}, \epsilon_{\Xi}\right)$ *in* (6.2) *is differentiable everywhere.*

Proof. The model $m_{\Xi_N}\left(\boldsymbol{x};\tilde{\boldsymbol{\beta}}_\Xi,\epsilon_\Xi\right)$ in (6.2) is differentiable everywhere if and only if $\phi_{\Xi_i}\left(\boldsymbol{x};\epsilon_\Xi\right)=\varphi_\Xi\left(\epsilon_\Xi\cdot\|\boldsymbol{x}-\boldsymbol{x}_i\|_2\right)$ is differentiable everywhere (see Proposition 2.1). If that is the case, then the composite function $\exp\left\{\chi_\Xi^{(1)}\cdot m_{\Xi_N}\left(\boldsymbol{x};\tilde{\boldsymbol{\beta}}_\Xi,\epsilon_\Xi\right)+\chi_\Xi^{(2)}\right\}$ is also differentiable everywhere. Furthermore:

$$1+\exp\left\{\chi_\Xi^{(1)}\cdot m_{\Xi_N}\left(\boldsymbol{x};\tilde{\boldsymbol{\beta}}_\Xi,\epsilon_\Xi\right)+\chi_\Xi^{(2)}\right\}\neq 0,\quad\forall\boldsymbol{x}\in\mathbb{R}^n.$$

Therefore, due to the reciprocal rule, $s_{\Xi_N}\left(\boldsymbol{x};\boldsymbol{\chi}_\Xi,\epsilon_\Xi\right)$ in (6.15) is differentiable everywhere. □

***Lemma 6.1: Gradient of the sigmoid in* (6.15).** *Suppose that the sigmoid $s_{\Xi_N}\left(\boldsymbol{x};\boldsymbol{\chi}_\Xi,\epsilon_\Xi\right)$ in* (6.15) *is differentiable everywhere with respect to $\boldsymbol{x}$ (see Proposition 6.3). Then, its gradient is:*

$$\nabla_{\boldsymbol{x}}s_{\Xi_N}\left(\boldsymbol{x};\boldsymbol{\chi}_\Xi,\epsilon_\Xi\right)=-\chi_\Xi^{(1)}\cdot s_{\Xi_N}\left(\boldsymbol{x};\boldsymbol{\chi}_\Xi,\epsilon_\Xi\right)\cdot\left[1-s_{\Xi_N}\left(\boldsymbol{x};\boldsymbol{\chi}_\Xi,\epsilon_\Xi\right)\right]\cdot\nabla_{\boldsymbol{x}}m_{\Xi_N}\left(\boldsymbol{x};\tilde{\boldsymbol{\beta}}_\Xi,\epsilon_\Xi\right),\tag{6.22}$$

where:

$$\nabla_{\boldsymbol{x}}m_{\Xi_N}\left(\boldsymbol{x};\tilde{\boldsymbol{\beta}}_\Xi,\epsilon_\Xi\right)=\sum_{i=1}^N\beta_\Xi^{(i)}\cdot\nabla_{\boldsymbol{x}}\phi_{\Xi_i}\left(\boldsymbol{x};\epsilon_\Xi\right).$$

Proof. We can compute the gradient of $s_{\Xi_N}\left(\boldsymbol{x};\boldsymbol{\chi}_\Xi,\epsilon_\Xi\right)$ in (6.15) by applying the chain rule:

$$\begin{aligned}\nabla_{\boldsymbol{x}}s_{\Xi_N}\left(\boldsymbol{x};\boldsymbol{\chi}_\Xi,\epsilon_\Xi\right)&=\frac{d}{dt}\frac{1}{1+\exp\left\{\chi_\Xi^{(1)}\cdot t+\chi_\Xi^{(2)}\right\}}\Bigg|_{t=m_{\Xi_N}\left(\boldsymbol{x};\tilde{\boldsymbol{\beta}}_\Xi,\epsilon_\Xi\right)}\cdot\nabla_{\boldsymbol{x}}m_{\Xi_N}\left(\boldsymbol{x};\tilde{\boldsymbol{\beta}}_\Xi,\epsilon_\Xi\right)\\&=-\chi_\Xi^{(1)}\cdot\frac{\exp\left\{\chi_\Xi^{(1)}\cdot t+\chi_\Xi^{(2)}\right\}}{\left[1+\exp\left\{\chi_\Xi^{(1)}\cdot t+\chi_\Xi^{(2)}\right\}\right]^2}\Bigg|_{t=m_{\Xi_N}\left(\boldsymbol{x};\tilde{\boldsymbol{\beta}}_\Xi,\epsilon_\Xi\right)}\cdot\nabla_{\boldsymbol{x}}m_{\Xi_N}\left(\boldsymbol{x};\tilde{\boldsymbol{\beta}}_\Xi,\epsilon_\Xi\right)\\&=-\chi_\Xi^{(1)}\cdot\frac{1}{1+\exp\{-\}}\cdot\left[1-\frac{1}{1+\exp\{-\}}\right]\Bigg|_{t=m_{\Xi_N}\left(\boldsymbol{x};\tilde{\boldsymbol{\beta}}_\Xi,\epsilon_\Xi\right)}\cdot\nabla_{\boldsymbol{x}}m_{\Xi_N}\left(\boldsymbol{x};\tilde{\boldsymbol{\beta}}_\Xi,\epsilon_\Xi\right)\\&=-\chi_\Xi^{(1)}\cdot s_{\Xi_N}\left(\boldsymbol{x};\boldsymbol{\chi}_\Xi,\epsilon_\Xi\right)\cdot\left[1-s_{\Xi_N}\left(\boldsymbol{x};\boldsymbol{\chi}_\Xi,\epsilon_\Xi\right)\right]\cdot\nabla_{\boldsymbol{x}}m_{\Xi_N}\left(\boldsymbol{x};\tilde{\boldsymbol{\beta}}_\Xi,\epsilon_\Xi\right).\end{aligned}$$

□

6.1.3 PSVMs for black-box and preference-based optimization

In the context of constrained BBO and PBO, $p_N\left(\boldsymbol{x}\in\Xi\right)$ in (6.15) is used to approximate the Ξ-feasible region of the GOP (2.1). New candidate samples $\boldsymbol{x}_{N+1}\in\Omega$ can be found by solving:

$$\boldsymbol{x}_{N+1}=\arg\max_{\boldsymbol{x}}a_N\left(\boldsymbol{x}\right)\tag{6.23}$$

$$\begin{aligned} \text{s.t.} \quad & \boldsymbol{x} \in \Omega \\ & p_N\left(\boldsymbol{x} \in \Xi\right) \geq \gamma, \end{aligned}$$

where $a_N\left(\boldsymbol{x}\right)$ is a suitable acquisition function, see for example Problem (2.72) and Problem (2.74). Assume to have performed several iterations of a BBO or PBO procedure, obtaining the sets $\mathcal{X}$ in (2.9) and $\mathcal{U}_\Xi$ in (2.11). Furthermore, suppose that $\mathcal{X}$ contains a combination of Ξ-feasible and Ξ-infeasible samples. When using PSVMs (or any classifier with a probabilistic interpretation) for black-box and preference-based optimization, a key issue might arise: *the current best candidate $\boldsymbol{x_{best}}\left(N\right)$ of the BBO or PBO method, which must be Ξ-feasible under the aforementioned assumptions, might be misclassified*. We can attribute this problem to (i) the soft margin formulation of the SVM classifier (see Section 6.1.1) and (ii) the maximum likelihood estimation in Problem (6.17), which treats all samples in $\mathcal{X}$ as "equals". If $\boldsymbol{x_{best}}\left(N\right)$ is misclassified, then $p_N\left(\boldsymbol{x_{best}}\left(N\right) \in \Xi\right) < \gamma$, making it less likely for a BBO or PBO procedure that looks for $\boldsymbol{x}_{N+1}$ by solving Problem (6.23) to explore in a neighborhood of $\boldsymbol{x_{best}}\left(N\right)$. In turn, this can lead the optimization method to miss a global (or even local) minimizer of the GOP (2.1). This issue is especially relevant when the global minimizer(s) are located on the boundary of Ξ of the GOP (2.1), in which case finding them with high accuracy can prove to be particularly hard. Figure 27 depicts an example of the previously described shortcoming of PSVMs.

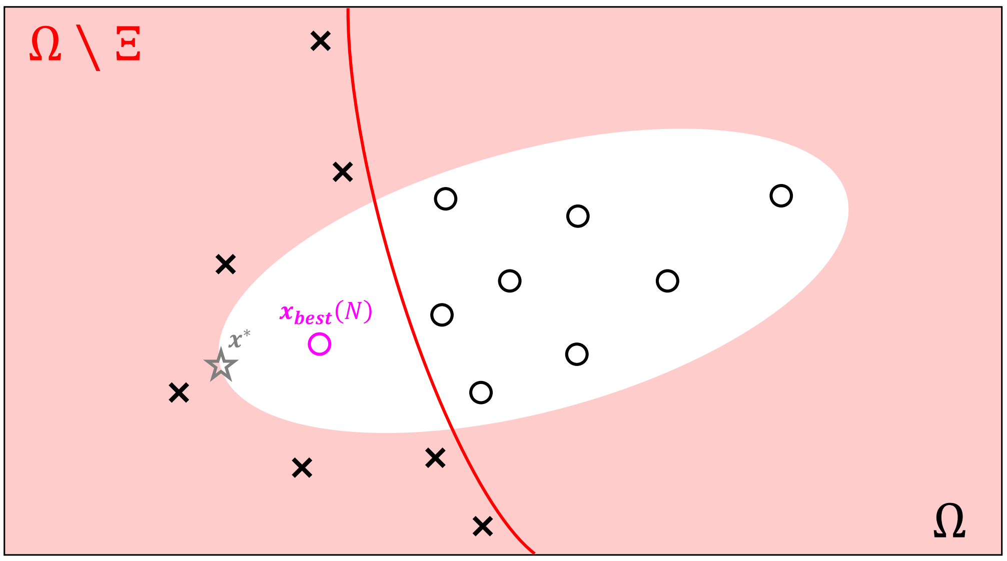


Figure 27: Representation of one of the issues that can affect BBO and PBO procedures which rely on a classifier to estimate the Ξ-feasible region of the GOP (2.1). The black circles denote the Ξ-feasible samples available to the optimization procedure whereas the black crosses are the Ξ-infeasible ones. The current best candidate is colored in magenta. The shaded red area depicts the Ξ-infeasible region of the GOP (2.1) while the red curve is the decision boundary of the classifier, trained from the samples at hand. Finally, the grey star is the global minimizer of the global optimization problem (which is yet to be found by the BBO or PBO procedure).

One way to make it more likely that $\boldsymbol{x_{best}}(N)$ is correctly classified is to weigh each sample differently in $J_{PSVM}(\chi_\Xi)$ in (6.16), such as[1]:

$$J_{PSVM}(\chi_\Xi) = -\sum_{i=1}^{N} \omega_i \cdot \left[u_i \cdot \ln s_{\Xi_N}(\boldsymbol{x}_i; \chi_\Xi, \epsilon_\Xi) + (1 - u_i) \cdot \ln\left(1 - s_{\Xi_N}(\boldsymbol{x}_i; \chi_\Xi, \epsilon_\Xi)\right)\right], \tag{6.24}$$

where $\omega_i \in \mathbb{R}_{\geq 0}, i = 1, \ldots, N$, are the weights associated to each sample $\boldsymbol{x}_i \in \mathcal{X}$. For example, we could associate higher ω_i's to those points that are closest to $\boldsymbol{x_{best}}(N)$, with a maximum at exactly the current best candidate, and decrease the weights as the distance from $\boldsymbol{x_{best}}(N)$ increases. However, we would still have no guarantee that $p_N(\boldsymbol{x_{best}}(N) \in \Xi) \geq \gamma$ for Problem (6.23). In this book, we propose a simple yet effective fix to the issue: *enforce the* Ξ*-feasibility of* $\boldsymbol{x_{best}}(N)$ *explicitly when solving Problem* (6.17). Formally, we find χ_Ξ for the sigmoid $s_{\Xi_N}(\boldsymbol{x}; \chi_\Xi, \epsilon_\Xi)$ in (6.15) by solving:

$$\begin{aligned} \arg\min_{\chi_\Xi} \; & J_{PSVM}(\chi_\Xi) \\ \text{s.t.} \quad & s_{\Xi_N}(\boldsymbol{x_{best}}(N); \chi_\Xi, \epsilon_\Xi) \geq \gamma \end{aligned} \tag{6.25}$$

instead of Problem (6.17). The constraint $s_{\Xi_N}(\boldsymbol{x_{best}}(N); \chi_\Xi, \epsilon_\Xi) \geq \gamma$ ensures that the current best candidate is deemed as Ξ-feasible by $\hat{u}_{\Xi_N}(\cdot)$ in (6.19). Moreover, the addition of the latter constraint does not compromise the convexity of Problem (6.25) (see Proposition 6.2), as claimed by the following Proposition.

Proposition 6.4. *Problem* (6.25) *is a convex nonlinear program.*

Proof. Both the cost function and the constraint function of Problem (6.25) are nonlinear in χ_Ξ, making it a nonlinear program. In Proposition 6.2, we have seen that $J_{PSVM}(\chi_\Xi)$ in (6.16) is a convex function. Hence, to prove the convexity of Problem (6.25), we only need to check if the function:

$$h(\chi_\Xi) = \gamma - s_{\Xi_N}(\boldsymbol{x_{best}}(N); \chi_\Xi, \epsilon_\Xi), \tag{6.26}$$

associated to the inequality constraint, is also convex with respect to χ_Ξ [18]. It is possible to prove that the sigmoid $s_{\Xi_N}(\boldsymbol{x_{best}}(N); \chi_\Xi, \epsilon_\Xi)$ in (6.15) is (logarithmically) concave with respect to χ_Ξ, see [96]. Then, $-s_{\Xi_N}(\boldsymbol{x_{best}}(N); \chi_\Xi, \epsilon_\Xi)$ must be convex, making the function $h(\chi_\Xi)$ in (6.26) convex as well. □

Problem (6.25) can be solved using derivative-based optimization procedures. The gradient and

[1]The rationale of minimizing the cost function in (6.24) is often referred to as *Maximum Weighted Likelihood Estimation (MWLE)*, see [74, 150]. Typically, MWLE is used for classification problems with highly unbalanced datasets, i.e. when the observations associated to either the positive or the negative class are scarce. In such cases, the weights ω_i in (6.24) assume statistical meaning and are defined based on prior information on the distributions of the two classes.

the Hessian of $J_{PSVM}\left(\chi_{\Xi}\right)$ are reported in (6.18), while the gradient of constraint function in (6.26) is:

$$\nabla_{\chi_{\Xi}}\left[\gamma - s_{\Xi_N}\left(\boldsymbol{x_{best}}\left(N\right);\chi_{\Xi},\epsilon_{\Xi}\right)\right] = \tag{6.27}$$
$$= \begin{bmatrix} m_{\Xi_N}\left(\boldsymbol{x_{best}}\left(N\right);\tilde{\boldsymbol{\beta}}_{\Xi},\epsilon_{\Xi}\right) \\ 1 \end{bmatrix} \cdot s_{\Xi_N}\left(\boldsymbol{x_{best}}\left(N\right);\chi_{\Xi},\epsilon_{\Xi}\right)\cdot\left[1 - s_{\Xi_N}\left(\boldsymbol{x_{best}}\left(N\right);\chi_{\Xi},\epsilon_{\Xi}\right)\right].$$

The expression in (6.27) can be derived similarly to the gradient in Lemma 6.1, by differentiating the sigmoid in (6.15) with respect to χ_{Ξ} instead of $\boldsymbol{x}$.
The example depicted in Figure 28 compares the decision boundary of the PSVM classifier (described in Section 6.1.2) against the one of the revisited classifier (proposed in this Section) . Notice how the former fails to guarantee the Ξ-feasibility of the best candidate $\boldsymbol{x_{best}}\left(N\right)$. In practice, this behavior depends quite a lot on the choice of the hyper-parameters of the classifier, namely the shape parameter ϵ_{Ξ}, the radial function $\varphi_{\Xi}\left(\cdot\right)$ and the trade-off weight C_{SVM}.

Remark 6.2 (What about IDWI?)**.** *`C-GLIS` and `C-GLISp` [156] use the inverse distance weighting interpolant to approximate the probability of Ξ-feasibility (see Definition 4.2). An advantage of the latter, as opposed to the traditional PSVMs in Section 6.1.2, is that $p_N\left(\boldsymbol{x}\in\Xi\right)$ in* (4.16) *is such that:*

$$p_N\left(\boldsymbol{x}_i\in\Xi\right) = 1, \quad \forall \boldsymbol{x}_i\in\mathcal{X} \text{ such that } u_i = 1, u_i\in\mathcal{U}_{\Xi}.$$

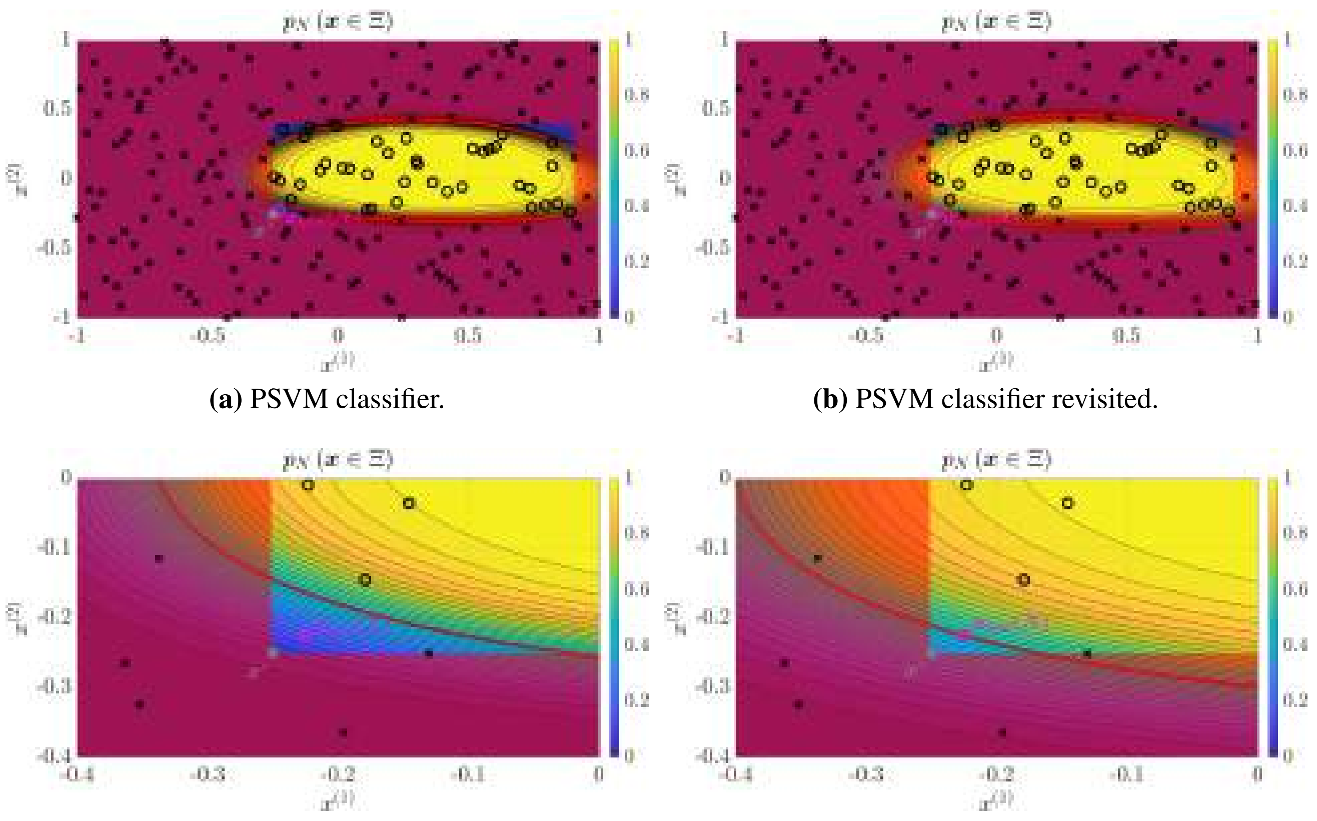
Hence, the best candidates of `C-GLIS` and `C-GLISp` [156] are always deemed as Ξ-feasible by the IDWI function. In practice, we have tested the performances of the proposed methods, `C-GLIS-r` and `C-GLISp-r`, when equipped with either the PSVM classifier described in this Section or the IDWI one in (4.16). *Empirically, as we will see in Chapter 7, we have found out that both $p_N\left(\boldsymbol{x}\in\Xi\right)$ in* (4.16) *and in* (6.15) *can perform well on different benchmark optimization problems and often result in similar performances. However, we argue that if noise were to be present during the evaluation of the black-box constraints (i.e. some of the $u_i\in\mathcal{U}_{\Xi}$ in* (2.11) *might be mislabeled), then PSVMs would be better suited for the task at hand as opposed to an interpolation method.*

We conclude this Section with some examples on the proposed classifier.

Example 6.1: Examples on the revisited PSVM classifier

Consider the four Ξ-feasible regions defined in Example 4.5. For each $\Xi_j, j = 1,\ldots,4$, we use the same $N_{init} = 200$ samples generated in the latter Example to train PSVM classifiers as proposed in this Section. We use the following hyper-parameters:

- $\epsilon_{\Xi} = 1$ and $\varphi_{\Xi}\left(\cdot\right)$ Gaussian for $m_{\Xi_N}\left(\cdot\right)$ in (6.2);



(**a**) PSVM classifier.

(**b**) PSVM classifier revisited.

(**c**) PSVM classifier (zoom).

(**d**) PSVM classifier revisited (zoom).

Figure 28: Comparison between the traditional PSVM classifier and the proposed one, tailored for constrained BBO and PBO. The decision boundaries are the red curves and are defined as in (6.20) **with** $\gamma = 0.5$**. When training the SVM classifier, we have chosen** $\varphi_\Xi(\cdot)$ **for** $m_{\Xi_N}\left(x; \tilde{\beta}_\Xi, \epsilon_\Xi\right)$ **in** (6.2) **to be a Gaussian RBF while the shape parameter is set to** $\epsilon_\Xi = 0.25$**. Furthermore,** $C_{SVM} = 10^6$ **for Problem** (6.8)**. The** Ξ**-feasible region (positive class) is** $\Xi = \left\{x : \begin{bmatrix}-0.25 & -0.25\end{bmatrix}^\top \leq x \leq \begin{bmatrix}0.9 & 0.4\end{bmatrix}^\top\right\}$**. The shaded red area denotes the** Ξ**-infeasible region. The classifiers are trained from** $N = 200$ **samples. We depict the** Ξ**-feasible ones with black circles, whereas the** Ξ**-infeasible ones are the black crosses. The current best candidate is highlighted in magenta. Finally, we assume that the global minimizer of the GOP** (2.1) **is located at** $\begin{bmatrix}0.25 & 0.25\end{bmatrix}^\top$**, as depicted by the grey star.**

- $C_{SVM} = 10^6$ for Problem (6.8);
- $\gamma = 0.5$ for the decision boundary of the PSVM classifier in (6.20).

Figure 29 depicts $p_N(x \in \Xi)$ for each Ξ-feasible region. Compared to the results obtained with IDWI, in Figure 15, the decision boundaries of the proposed PSVMs are more similar to the boundaries of the sets Ξ_j, $j = 1, \ldots, 4$.

6.1.4 Recalibration of the classifier's hyper-parameters

There are three tuning knobs for the PSVM classifier described in Section 6.1.2: the radial function $\varphi_\Xi(\cdot)$ and the shape parameter ϵ_Ξ for $m_{\Xi_N}(\cdot)$ in (6.2), but also the trade-off weight C_{SVM} of Problem

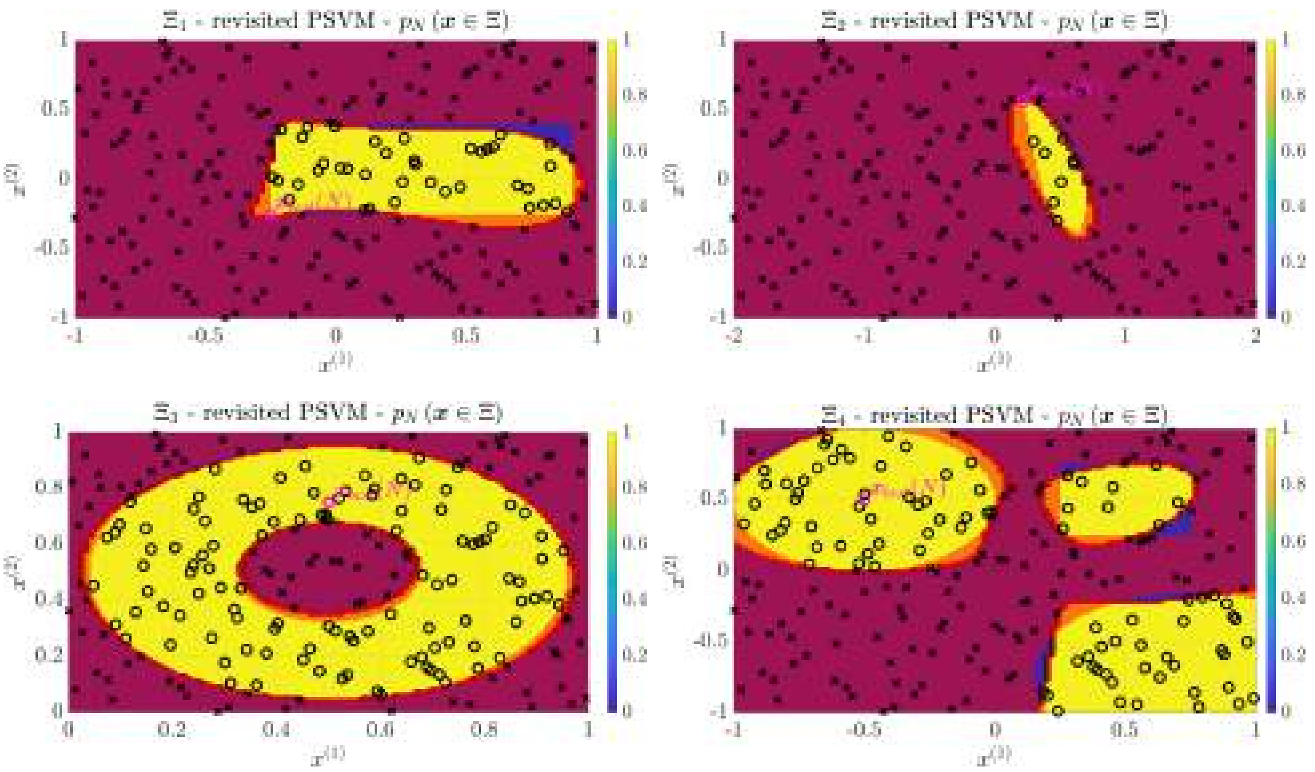


Figure 29: Probability of Ξ-feasibility estimated by the revisited PSVM classifier for different Ξ-feasible regions, defined in Example 6.1. The Ξ-feasible samples are depicted as black circles whereas the Ξ-infeasible ones are black crosses. The current best candidate is colored in magenta. The shaded red areas denote the Ξ-infeasible regions. Finally, the red lines are the decision boundaries, i.e. $p_N(x \in \Xi) = \gamma$, with $\gamma = 0.5$.

(6.8). In practice, we could select them in a similar fashion to Section 4.4.2, where we have employed K-fold grid search cross-validation [58] to recalibrate the surrogate model of the cost function of the GOP (2.1). However, at this point of the book, we have a much better alternative: algorithm `GLIS-r` [108].

In what follows, we do not consider the recalibration of the radial function $\varphi_\Xi(\cdot)$, although it could be handled by adding a discrete decision variable that can only assume a finite number of values (one for each possible choice). Thus, we define the decision vector for the purpose of recalibrating the hyper-parameters of the PSVM classifier as:

$$\boldsymbol{x}_\Xi = \begin{bmatrix} \epsilon_\Xi & C_{SVM} \end{bmatrix}^\top, \quad \boldsymbol{x}_\Xi \in \mathbb{R}^2_{>0}. \tag{6.28}$$

We are interested in *minimizing the out-of-sample misclassification rate*, which is a common measure of performance for classifiers. The latter can be estimated by means of K-fold cross-validation [58]. Hence, we can build a suitable black-box cost function for the optimization of $\boldsymbol{x}_\Xi$ in (6.28) as follows:

1. Build the dataset:

$$\mathcal{D} = \{(\boldsymbol{x}_i, u_i) : \boldsymbol{x}_i \in \mathcal{X}, u_i \in \mathcal{U}_\Xi, i = 1, \dots, N\} \tag{6.29}$$

from the data at hand (see (2.9) and (2.11)).

2. Remove the entry in $\mathcal{D}$ (6.29) that is associated to $\boldsymbol{x_{best}}(N)$:

$$\mathcal{D} = \mathcal{D} \setminus \{(\boldsymbol{x_{best}}(N), 1)\}. \tag{6.30}$$

Note that it only makes sense to train a classifier when $\mathcal{X}$ in (2.9) contains both Ξ-feasible and Ξ-infeasible samples, hence $u_\Xi(\boldsymbol{x_{best}}(N)) = 1$. Furthermore, the current best candidate must always be present in the training set for the revisited PSVM in Section 6.1.3, due to how we estimate the probability of Ξ-feasibility for the proposed classifier (see Problem (6.25)). For this reason, we remove $(\boldsymbol{x_{best}}(N), 1)$ from $\mathcal{D}$ in (6.29) and add it back when needed, as we will see shortly. Thus, the cardinality of $\mathcal{D}$ in (6.30) is $N_\mathcal{D} = N - 1$.

3. Choose a ratio $R_\Xi \in [0, 1]$ and compute the maximum number of elements for each fold as:

$$N_{\mathcal{D}_{fold}} = \max\{\lfloor R_\Xi \cdot N_\mathcal{D} \rfloor, 1\}.$$

The resulting number of folds is:

$$K = \left\lceil \frac{N_\mathcal{D}}{N_{\mathcal{D}_{fold}}} \right\rceil.$$

4. Randomly partition $\mathcal{D}$ into K (almost) equally-sized subsets (folds) $\mathcal{D}^{(1)}, \dots, \mathcal{D}^{(K)}$, i.e. such that:

$$\mathcal{D} = \bigcup_{k=1}^{K} \mathcal{D}^{(k)}, \quad \mathcal{D}^{(i)} \cap \mathcal{D}^{(j)} = \emptyset, \forall i, j = 1, ..., K, i \neq j$$

and $\left|\mathcal{D}^{(k)}\right| \leq N_{\mathcal{D}_{fold}}, \forall k = 1, \dots, K$. In particular, we suggest using *stratified* K-fold cross-validation [20], i.e. the dataset $\mathcal{D}$ is partitioned into K folds such that each class is uniformly distributed among all subsets of $\mathcal{D}$.

5. Generate all possible *training sets*:

$$\mathcal{D}_{train}^{(k)} = \left[\bigcup_{j \neq k} \mathcal{D}^{(j)}\right] \cup \{(\boldsymbol{x_{best}}(N), 1)\}, \quad k = 1, \dots, K,$$

and *validation sets*:

$$\mathcal{D}_{val}^{(k)} = \mathcal{D}^{(k)}, \quad k = 1, \dots, K.$$

6. The cost function for the recalibration of $\boldsymbol{x}_\Xi$ in (6.28) is computed as follows. Consider a decision vector $\tilde{\boldsymbol{x}}_\Xi = \begin{bmatrix} \tilde{\epsilon}_\Xi & \tilde{C}_{SVM} \end{bmatrix}^\top \in \mathbb{R}^2_{>0}$, then:

   a) Train K PSVM classifiers with hyper-parameters $\tilde{\epsilon}_\Xi$ and $\tilde{C}_{SVM}$ as proposed in Section 6.1.3, starting from the training sets $\mathcal{D}_{train}^{(k)}, k = 1, \dots, K$;

b) Build the surrogate Ξ-feasibility functions $\hat{u}^{(k)}_{\Xi_{\tilde{N}}}(\cdot), k = 1, \dots, K$, in (6.19), where $\tilde{N} = \left|\mathcal{D}^{(k)}_{train}\right|$;

c) Compute the misclassification rates of each classifier on the validation sets:

$$J_{R_\Xi}\left(\hat{u}^{(k)}_{\Xi_{\tilde{N}}}(\cdot), \mathcal{D}^{(k)}_{val}\right) = \sum_{(\boldsymbol{x}_i, u_i) \in \mathcal{D}^{(k)}_{val}} \frac{\mathbb{I}\left(\hat{u}^{(k)}_{\Xi_{\tilde{N}}}(\boldsymbol{x}_i) \neq u_i\right)}{\left|\mathcal{D}^{(k)}_{val}\right|}, \quad k = 1, \dots, K. \tag{6.31}$$

d) The value of the black-box cost function associated to the decision vector $\tilde{\boldsymbol{x}}_\Xi$ is the *average out-of-sample misclassification rate*:

$$f(\tilde{\boldsymbol{x}}_\Xi) = \frac{\sum_{k=1}^{K} J_{R_\Xi}\left(\hat{u}^{(k)}_{\Xi_{\tilde{N}}}(\cdot), \mathcal{D}^{(k)}_{val}\right)}{K}. \tag{6.32}$$

`GLIS-r` [108] (Algorithm 14) evaluates $N_{max} \in \mathbb{N}$ different calibrations of $\boldsymbol{x}_\Xi$ in (6.28) and returns the one which achieves the lowest value of $f(\boldsymbol{x}_\Xi)$ in (6.32). The same training sets $\mathcal{D}^{(k)}_{train}$ and validation sets $\mathcal{D}^{(k)}_{val}$, $k = 1, \dots, K$, are used throughout the whole optimization process, hence they need to be generated only once. We impose box constraints on the hyper-parameters of the PSVM classifier:

$$\Omega_\Xi = \{\boldsymbol{x}_\Xi : \boldsymbol{l}_\Xi \leq \boldsymbol{x}_\Xi \leq \boldsymbol{u}_\Xi\},$$

where $\boldsymbol{l}_\Xi, \boldsymbol{u}_\Xi \in \mathbb{R}^2_{>0}, \boldsymbol{l}_\Xi \leq \boldsymbol{u}_\Xi$. The corresponding *unconstrained black-box optimization problem*, solved by `GLIS-r` [108], is:

$$\begin{aligned} &\arg\min_{\boldsymbol{x}_\Xi} f(\boldsymbol{x}_\Xi) \\ &\text{s.t.} \quad \boldsymbol{x}_\Xi \in \Omega_\Xi. \end{aligned} \tag{6.33}$$

Typically, ϵ_Ξ and C_{SVM} can span multiple orders of magnitudes, hence it is better to consider $\boldsymbol{x}_\Xi = \begin{bmatrix}\log_{10}\epsilon_\Xi & \log_{10} C_{SVM}\end{bmatrix}^\top$ instead of (6.28).

Remark 6.3 (On the computational burden of the recalibration procedure)**.** *`GLIS-r` [108] (approximately) solves Problem* (6.33) *by performing N_{max} sample evaluations, resulting in the estimation of $N_{max} \cdot K$ surrogate Ξ-feasibility functions in* (6.19)*. Alternatively, to reduce the computational burden, we could employ hold-out validation [61] instead of K-fold cross-validation. $\mathcal{D}$ in* (6.30) *is simply split into two subsets, the training set $\mathcal{D}_{train}$, containing $N_{train} = (N-1) \cdot R_\Xi$ elements, and the validation set $\mathcal{D}_{val}$, composed of $N_{val} = N - 1 - N_{train}$ entries. When hold-out validation is employed for the recalibration of $\boldsymbol{x}_\Xi$ in* (6.28)*, the PSVM classifier is only trained N_{max} times, reducing the computational overhead, but the estimates of the out-of-sample misclassification rates might be less accurate [61].*

Example 6.2: Recalibration of PSVM by means of GLIS-r [108]

Consider the black-box constraint set:

$$\Xi = \left\{ \boldsymbol{x} : \|\boldsymbol{x} - \boldsymbol{x}_{c_1}\|_2^2 \leq r_1^2 \right\} \cup \left\{ \boldsymbol{x} : \|\boldsymbol{x} - \boldsymbol{x}_{c_2}\|_2^2 \leq r_2^2 \right\} \cup \{\boldsymbol{x} : 0.25 \cdot \mathbf{1}_2 \leq \boldsymbol{x} \leq 0.75 \cdot \mathbf{1}_2\},$$

where $\boldsymbol{x}_{c_1} = \begin{bmatrix} -0.5 & 0.5 \end{bmatrix}^\top$, $r_1 = 0.5$, $\boldsymbol{x}_{c_2} = \begin{bmatrix} 0.75 & -0.75 \end{bmatrix}^\top$ and $r_2 = 0.6$. We generate a set $\mathcal{X}$ (2.9) of $N = 100$ samples, using a latin hypercube design (see Section 2.4), and evaluate the Ξ-feasibility of each point, obtaining the set $\mathcal{U}_\Xi$ in (2.11). Then, we calibrate the hyper-parameters of the PSVM classifier described in Section 6.1.3. For the sake of simplicity, we only tune the shape parameter ϵ_Ξ for the model $m_{\Xi_N}(\cdot)$ in (6.2). The radial function $\varphi_\Xi(\cdot)$ is Gaussian whereas the last hyper-parameter is set to $C_{SVM} = 10^6$. Hence, $x_\Xi = \epsilon_\Xi$ and, in particular, we consider $l_\Xi = 10^{-3}$ and $u_\Xi = 10$. Lastly, the threshold γ for the decision boundary in (6.20) is set to $\gamma = 0.5$.

We employ GLIS-r [108] (Algorithm 14) to solve Problem (6.33) as proposed in this Section. We use the following hyper-parameters for GLIS-r [108]:

- Number of initial samples $N_{init} = 6$,
- Budget $N_{max} = 30$,
- Hyper-parameters for the surrogate model $\hat{f}_{N_\Xi}(\boldsymbol{x}_\Xi)$ in (4.1): $\epsilon_f = 1$, $\varphi_f(\cdot)$ inverse quadratic, $\epsilon_{SVD} = 10^{-6}$,
- Cycling set $\Delta_{cycle} = \langle 0.95, 0.7, 0.35, 0 \rangle$,
- Number of clusters for $\mathcal{X}_{aug}$: $K_{aug} = 5$,
- No recalibration of $\hat{f}_{N_\Xi}(\boldsymbol{x}_\Xi)$ in (4.1), i.e. $\mathcal{K}_{R_f} = \emptyset$, $\mathcal{M}_f = \emptyset$.

Problem (5.13) is solved by the PSWARM [72] algorithm (see Section 1.2.5). Figure 30 shows the performances of GLIS-r [108] and compares the initial PSVM classifier ($\epsilon_\Xi = 0.1$, $f(\cdot) = 0.26$) to the recalibrated one ($\epsilon_\Xi \approx 1.8$, $f(\cdot) = 0.05$). To avoid confusion, we refer to the samples related to Problem (6.33) using the subscript Ξ; similarly for their number, which is N_Ξ. Notice how the recalibrated PSVM classifier is able to identify the three disconnected regions which constitute Ξ, whereas the initial one is not successful in doing so.

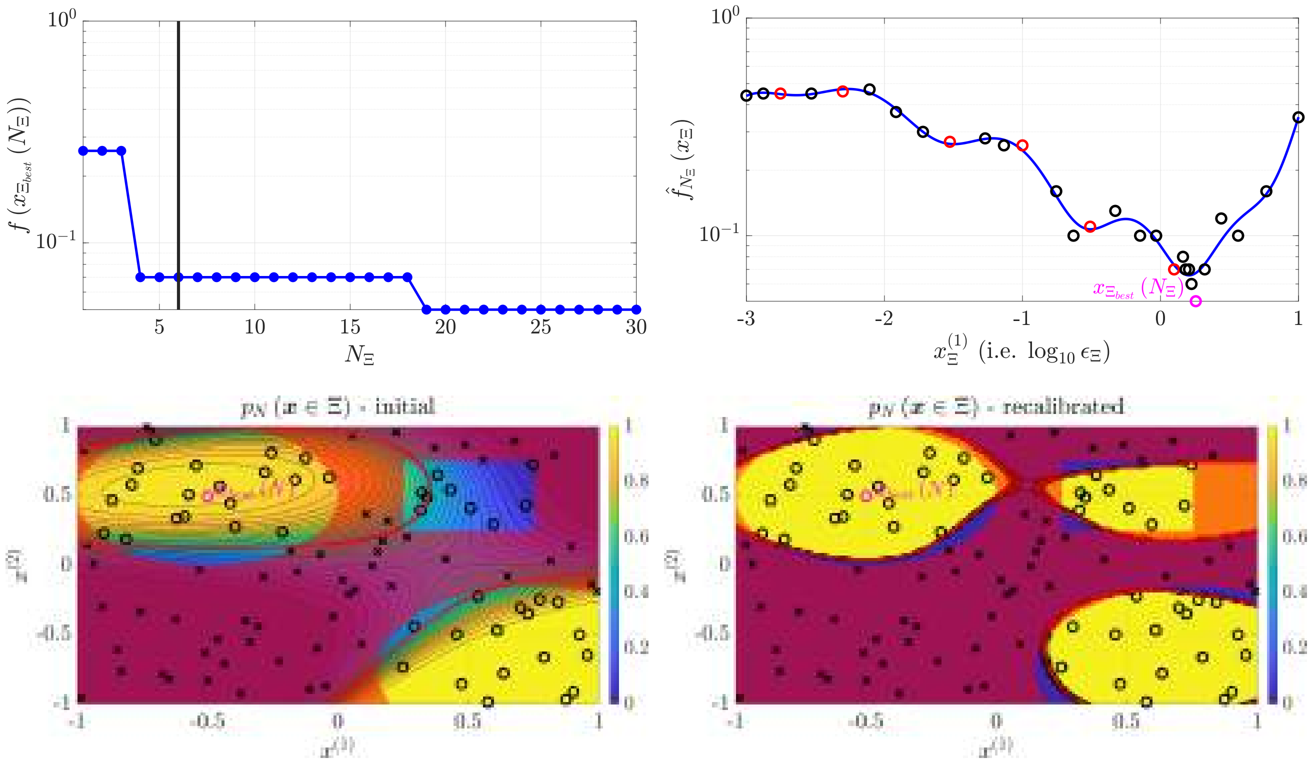


Figure 30: Results obtained in Example 6.2. On the top left: performances of the `GLIS-r` [108] algorithm. On the top right: surrogate model of the cost function in (6.32) **for** $N_\Xi = N_{max} - 1 = 29$ **(i.e. at the last iteration of the optimization procedure). The tried calibrations are represented as circles; in particular, the red ones are obtained from the initial experimental design whereas the black ones are found by solving Problem** (5.13)**. The best calibration is highlighted in magenta. Notice how** $\hat{f}_{N_\Xi}\left(x_\Xi\right)$ **is not able to interpolate all the calibrations due to the erratic behavior of the average misclassification rate. On the bottom row: comparison of the initial PSVM classifier against the recalibrated one. The decision boundaries are the red curves and are defined as in** (6.20)**, with** $\gamma = 0.5$**. The** Ξ**-feasible samples in** $\mathcal{X}$ **are depicted with black circles whereas the** Ξ**-infeasible ones are the black crosses. The shaded red area denotes the** Ξ**-infeasible region. The current best candidate is colored in magenta.**

6.2 Infill sampling criterion

Now, we cover the infill sampling criterion employed by algorithms `C-GLIS-r` and `C-GLISp-r` when looking for new candidate samples. *We make no assumptions on the samples obtained by the initial experimental design, which could all be* Ξ*-infeasible.* Similarly to `COBRA` [112], we use different strategies depending on whether a Ξ-feasible sample is available or not. Suppose to be at the k-th iteration of either the `C-GLIS-r` or the `C-GLISp-r` procedure. We have at our disposal the set of samples $\mathcal{X}$, $|\mathcal{X}| = N$, in (2.9), the sets which contain the information on $f\left(\boldsymbol{x}\right)$ of the GOP (2.1), namely $\mathcal{Y}$ (BBO) or $\mathcal{B}$ and $\mathcal{S}$ (PBO) in (2.10), (3.9) and (3.10) respectively, and the Ξ-feasibility data in $\mathcal{U}_\Xi$ (2.11). We distinguish three possible situations.

No Ξ**-feasible samples are available (**$\mathcal{X} \cap \Xi = \emptyset$**).** In this case, there is no need to build the surrogate model for $f\left(\boldsymbol{x}\right)$ since we still have not found a point belonging to the Ξ-feasible region of the GOP

(2.1). At the same time, we are unable to train a PSVM classifier (as described in Section 6.1.3) because we only have at our disposal samples belonging to one class. We are left with only one option: *explore Ω in the hope of finding a Ξ-feasible sample*. To do so, we use the IDW distance function $z_N(\boldsymbol{x})$ in (4.17) to drive the search. We look for new candidate samples $\boldsymbol{x}_{N+1} \in \Omega$ as:

$$\begin{aligned} \boldsymbol{x}_{N+1} = \arg\min_{\boldsymbol{x}} & \; z_N(\boldsymbol{x}) \\ \text{s.t.} & \quad \boldsymbol{x} \in \Omega. \end{aligned} \tag{6.34}$$

We have already seen how performing pure exploration can be quite inefficient, at least when it comes to looking for the global minimizer(s) of the GOP (2.1), see Example 5.2. However, we have no other alternatives if we assume the black-box constraints functions to not be measurable. Furthermore, if we were to use the information on the cost function $f(\boldsymbol{x})$ to drive the search, we could mislead the optimization procedure into searching in those regions of Ω that are likely to have lower costs but with no guarantee of Ξ-feasibility. For example, in the context of control systems, aggressive controller tunings might lead to good closed-loop performances (low $f(\cdot)$) at the cost of stressing the actuators too much (Ξ-infeasibility). PBO is a bit of an exception: if the same decision-maker were to both express the preferences (information on $f(\boldsymbol{x})$) and assess the Ξ-feasibility (decision-maker-based constraint), then he/she is likely to penalize those calibrations that are Ξ-infeasible, resulting in higher scores (higher $f(\cdot)$'s).

The next Example shows the performances of the proposed strategy (described by Problem (6.34)) for different dimensionalities, n, of the decision vector. We also give some insights on how to improve the performances in case the measures of the black-box constraints functions are available (i.e. when the set $\mathcal{C}_\Xi$ in (2.12) is available).

Example 6.3: Pure exploration for finding Ξ-feasible samples

Consider a GOP (2.1) with feasible region described by the following sets:

$$\Omega = \{\boldsymbol{x} : \boldsymbol{0}_n \leq \boldsymbol{x} \leq \boldsymbol{1}_n\},$$
$$\Xi = \{\boldsymbol{x} : g_\Xi(\boldsymbol{x}) \leq 0\},$$

where:

$$g_\Xi(\boldsymbol{x}) = \|\boldsymbol{x} - \boldsymbol{x}_c\|_2 - r,$$

$\boldsymbol{x}_c = \frac{3}{4} \cdot \boldsymbol{1}_n$ and $r = 0.25$.

We want to assess the number of samples required by the proposed infill sampling criterion (Problem (6.34)) to find a Ξ-feasible sample, starting from only one sample,

$$\mathcal{X} = \{\boldsymbol{x}_1\}, \boldsymbol{x}_1 = \frac{1}{4} \cdot \mathbf{1}_n,$$

such that $\boldsymbol{x}_1 \notin \Xi$ and for different dimensionalities

$$n \in \{1, 2, 3, 5, 8, 10\}.$$

We compare the performances of the pure exploration approach (Problem (6.34)) to another one which uses the (noiseless) measures of the black-box constraint function in:

$$C_\Xi = \{c_i : c_i = g_\Xi(\boldsymbol{x}_i), \boldsymbol{x}_i \in \mathcal{X}\}.$$

The latter method operates as follows. At each iteration:

1. Estimate the surrogate model $\hat{g}_{\Xi_N}(\boldsymbol{x})$ of $g_\Xi(\boldsymbol{x})$ as proposed in Section 4.1.1. In particular, we have used $\varphi_{g_\Xi}(\cdot)$ inverse quadratic, $\epsilon_{g_\Xi} = 1$ and $\epsilon_{SVD} = 10^{-6}$.
2. Build the surrogate quadratic penalty function $\hat{\rho}_N : \mathbb{R}^n \to \mathbb{R}_{\geq 0}$ as in (1.21) but with $\hat{g}_{\Xi_N}(\boldsymbol{x})$ instead of $g_\Xi(\boldsymbol{x})$, i.e.:

$$\hat{\rho}_N(\boldsymbol{x}) = \left(\max\left\{0, \hat{g}_{\Xi_N}(\boldsymbol{x})\right\}\right)^2.$$

3. Look for a new candidate sample by solving the following global optimization problem:

$$\begin{aligned} \boldsymbol{x}_{N+1} = \arg\min_{\boldsymbol{x}} & \left[\delta \cdot \hat{\bar{\rho}}_N(\boldsymbol{x}; \mathcal{X}_{aug}) + (1-\delta) \cdot \bar{z}_N(\boldsymbol{x}; \mathcal{X}_{aug})\right] \\ \text{s.t.} \quad & \boldsymbol{x} \in \Omega. \end{aligned} \tag{6.35}$$

The acquisition function of Problem (6.35) is the same as the one of `GLIS-r` [108] and `GLISp-r` [109], in (5.12), but with the surrogate quadratic penalty function instead of the surrogate model of $f(\boldsymbol{x})$. In particular, for the sake of simplicity, we have fixed $\delta = 0.65$ (no greedy δ-cycling). Lastly, $K_{aug} = 5$ for the generation of $\mathcal{X}_{aug}$ as in Algorithm 13.

We compare the two approaches with respect to the number of samples required to find a Ξ-feasible point. In particular, in this Example, we can easily compute the minimum distance between a Ξ-feasible point and its closest sample in $\mathcal{X}$ (2.9):

$$\begin{aligned} \min_{\boldsymbol{x}_i \in \mathcal{X}} \min_{\boldsymbol{x} \in \Xi} \|\boldsymbol{x}_i - \boldsymbol{x}\|_2 &= \min_{\boldsymbol{x}_i \in \mathcal{X}} \max\{\|\boldsymbol{x}_i - \boldsymbol{x}_c\|_2 - r, 0\} \\ &= \min_{c_i \in C_\Xi} \max\{c_i, 0\}. \end{aligned}$$

We use both methods to generate 99 additional samples, obtaining a total of $N_{max} = 100$ points. Figure 31 depicts the results. Notice how the approach based on the surrogate quadratic penalty function (Problem (6.35)) is able to find a Ξ-feasible sample quite faster than its IDW-based counterpart (Problem (6.34)). Intuitively, higher-dimensional problems require more points for that purpose. In particular, as n increases, the ratio between the volumes of Ξ (hypersphere) and Ω (hypercube) drastically decreases. Hence, it becomes progressively harder to find a Ξ-feasible sample as the dimensionality of the problem increases. Lastly, we point out that the IDW-based approach tends to be more space-filling in Ω while the penalty one focuses more on a neighborhood of the Ξ-feasible region.

Only Ξ-feasible samples are available ($\mathcal{X} \cap \Xi = \mathcal{X}$). Whenever only Ξ-feasible samples are available, we simply proceed as we would in the unconstrained BBO or PBO framework. In particular, we adopt the infill sampling criterion of `GLIS-r` [108] and `GLISp-r` [109], namely we look for new candidate samples by solving the following optimization problem:

$$\begin{aligned} \boldsymbol{x}_{N+1} = \arg\min_{\boldsymbol{x}} & \; a_N(\boldsymbol{x}) \\ \text{s.t.} & \quad \boldsymbol{x} \in \Omega, \end{aligned} \tag{6.36}$$

where $a_N(\boldsymbol{x})$ is defined as in (5.12), i.e.:

$$a_N(\boldsymbol{x}) = \delta \cdot \hat{\bar{f}}_N\left(\boldsymbol{x}; \mathcal{X}_{aug}\right) + (1-\delta) \cdot \bar{z}_N\left(\boldsymbol{x}; \mathcal{X}_{aug}\right),$$

and δ cycled following the δ-cycling strategy (Section 5.2.3).

Both Ξ-feasible and Ξ-infeasible samples are available ($\emptyset \subset \mathcal{X} \cap \Xi \subset \mathcal{X}$). In this case, we must still *look for new candidate samples by trading off exploration and exploitation but, at the same time, penalize the search in those zones of Ω that are likely to contain Ξ-infeasible points*. To do so, we rely on: (i) the surrogate model $\hat{f}_N(\boldsymbol{x})$ in (4.1) (Section 4.1), (ii) the IDW distance function $z_N(\boldsymbol{x})$ in (4.17) (Section 4.2) and (iii) the probability of Ξ-feasibility obtained by the PSVM classifier $p_N(\boldsymbol{x} \in \Xi)$ in (6.15) (Section 6.1.3). We propose to look for new candidate samples by solving the

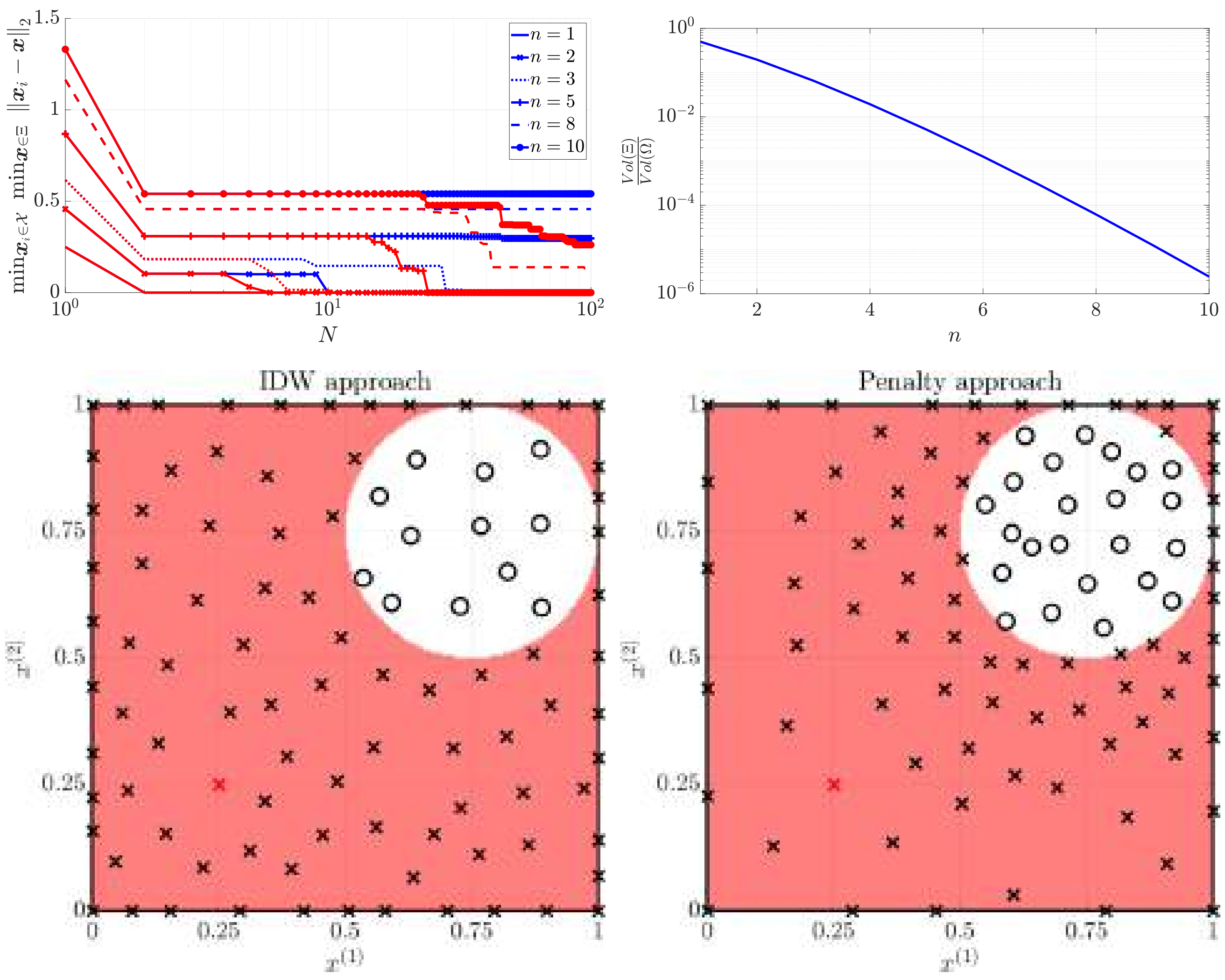


Figure 31: Results obtained in Example 6.3. On the top left: minimum distance between a Ξ-feasible point and its closest sample in $\mathcal{X}$ as $N = |\mathcal{X}|$ increases and for different n's. The IDW approach (Problem (6.34)) is depicted in blue while the penalty method (Problem (6.35)) is shown in red. On the top right: ratio of the volumes of Ξ (hypersphere) and Ω (hypercube) as n increases. On the bottom row: samples obtained by the two approaches when $n = 2$. The Ξ-feasible samples in $\mathcal{X}$ are depicted with black circles whereas the Ξ-infeasible ones are the black crosses. The initial sample x_1 is the red cross. The shaded red area denotes the Ξ-infeasible region.

following optimization problem:

$$
\begin{aligned}
\boldsymbol{x}_{N+1} = \arg \min_{\boldsymbol{x}, \varepsilon_\Xi} & \; a_N(\boldsymbol{x}, \varepsilon_\Xi) \\
\text{s.t.} \quad & \boldsymbol{x} \in \Omega \\
& p_N(\boldsymbol{x} \in \Xi) \geq \gamma \cdot (1 - \varepsilon_\Xi) \\
& 0 \leq \varepsilon_\Xi \leq 1,
\end{aligned} \tag{6.37}
$$

where the acquisition function $a_N : \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}$ is defined as (cf. (5.12)):

$$
a_N(\boldsymbol{x}, \varepsilon_\Xi) = \delta \cdot \hat{\bar{f}}_N(\boldsymbol{x}; \mathcal{X}_{aug}) + (1 - \delta) \cdot \bar{z}_N(\boldsymbol{x}; \mathcal{X}_{aug}) + \varepsilon_\Xi. \tag{6.38}
$$

The rationale behind Problem (6.37) is as follows. We drive the search towards those regions of Ω that are likely to contain Ξ-feasible points by including the constraint:

$$p_N(\boldsymbol{x} \in \Xi) \geq \gamma, \tag{6.39}$$

which is consistent with the surrogate Ξ-feasibility function $\hat{u}_{\Xi_N}(\boldsymbol{x})$ in (6.19). To take into account that the decision boundary of the PSVM classifier, $p_N(\boldsymbol{x} \in \Xi) = \gamma$, is only an approximation of the boundary of the set Ξ of the GOP (2.1), we add a *slack variable* $\varepsilon_\Xi \in [0, 1]$ and re-write the constraint in (6.39) as:

$$p_N(\boldsymbol{x} \in \Xi) \geq \gamma \cdot (1 - \varepsilon_\Xi). \tag{6.40}$$

In particular:

- If $\varepsilon_\Xi = 0$, then the constraint in (6.40) amounts to the one in (6.39);
- If $\varepsilon_\Xi = 1$, then the constraint in (6.40) becomes:

  $$p_N(\boldsymbol{x} \in \Xi) \geq 0,$$

  which is satisfied $\forall \boldsymbol{x} \in \mathbb{R}^n$. Therefore, in this case, Problem (6.37) is equivalent to Problem (6.36) and is not concerned with the set Ξ of the GOP (2.1).

To penalize the violation of the constraint in (6.39), we add the slack ε_Ξ to the acquisition function in (6.38). Instead, the first two terms of $a_N(\boldsymbol{x}, \varepsilon_\Xi)$ in (6.38), which match the ones of the acquisition function of GLIS-r [108] and GLISp-r [109] in (5.12), address the exploration-exploitation dilemma. Note that, due to min-max rescaling (Section 5.2.1), $\hat{\bar{f}}_N(\boldsymbol{x}; \mathcal{X}_{aug})$ and $\bar{z}_N(\boldsymbol{x}; \mathcal{X}_{aug})$ roughly assume the range $[0, 1]$ and are therefore comparable to ε_Ξ, which shares those bounds.

The definition of Problem (6.37) has been inspired by the infill sampling criteria of several BBO algorithms. First of all, the constraint in (6.39) is used by algorithms SuperEGO [125] and Constrained EGO with Support Vector Machines [9] (see Problem (2.72) and Problem (2.74)) to penalize the exploration in those zones of Ω which are likely to contain Ξ-infeasible samples. Next, let us consider the COBRA [112] algorithm, which assumes that all the black-box constraints functions, i.e. $g_\Xi^{(j)}(\boldsymbol{x}), j = 1, \ldots, q_\Xi$, in (2.2), are measurable. The latter method builds a surrogate model $\hat{g}_{\Xi_N}^{(j)}(\boldsymbol{x})$ (defined as in (2.29)) for each of them, as proposed in Section 2.5.1. Problem (2.63) describes the infill sampling strategy of COBRA [112] when $\emptyset \subset \mathcal{X} \cap \Xi \subset \mathcal{X}$; the latter includes the constraints:

$$\hat{g}_{\Xi_N}^{(j)}(\boldsymbol{x}) + \varsigma_\Xi^{(j)} \leq 0, \quad j = 1, \ldots, q_\Xi. \tag{6.41}$$

In the previous expression, $\varsigma_{\Xi}^{(j)}$ is a positive margin that is increased or decreased based on the number of subsequent successful or failed iterations (in a sense that they lead to Ξ-feasible or Ξ-infeasible samples)[2]. In practice, the slack ε_{Ξ} of Problem (6.37) (C-GLIS-r and C-GLISp-r) and the margins $\varsigma_{\Xi}^{(j)}, j = 1, \dots, q_{\Xi}$, of Problem (2.63) (COBRA [112]) constitute two different ways of addressing a common problem: the set Ξ of the GOP (2.1) is not available, we can only approximate it in some way. In particular, we approximate Ξ either as:

$$\hat{\Xi} = \{\boldsymbol{x} : p_N (\boldsymbol{x} \in \Xi) \geq \gamma\}, \tag{6.42}$$

in the former case, or:

$$\hat{\Xi} = \left\{\boldsymbol{x} : \hat{g}_{\Xi_N}^{(j)} (\boldsymbol{x}) \leq 0, j = 1, \dots, q_{\Xi}\right\}, \tag{6.43}$$

in the latter case. COBRA [112] is more conservative in a sense that it tends to sample far away from the boundary of $\hat{\Xi}$ in (6.43) due to the constraints in (6.41). Instead, C-GLIS-r and C-GLISp-r allow sampling outside $\hat{\Xi}$ in (6.42), when $0 < \varepsilon_{\Xi} \leq 1$ for Problem (6.37). We argue that the proposed slack formulation, albeit less conservative, can be helpful whenever the global minimizer(s) of the GOP (2.1) are on the boundary of Ξ. Roughly speaking, *the proposed infill sampling criterion allows us to get closer to the global minimizer(s) of the considered global optimization problem at the cost (if needed) of a sufficiently small violation of the constraint in* (6.39). More formally, suppose that the current best candidate $\boldsymbol{x_{best}} (N) \in \Xi$ of either C-GLIS-r or C-GLISp-r is quite close to a minimizer $\boldsymbol{x}_i^* \in \mathcal{X}^*$ of the GOP (2.1), i.e. $\left\|\boldsymbol{x_{best}} (N) - \boldsymbol{x}_i^*\right\|_2 < \epsilon$ with $\epsilon \in \mathbb{R}_{>0}$ small. We prove that, in some cases, Problem (6.37) can return a sample

$$\boldsymbol{x}_{N+1} \in \mathcal{B} (\boldsymbol{x_{best}} (N) ; \epsilon) \cap \Omega \cap \Xi$$

such that $\left\|\boldsymbol{x}_{N+1} - \boldsymbol{x}_i^*\right\|_2 < \left\|\boldsymbol{x_{best}} (N) - \boldsymbol{x}_i^*\right\|_2 < \epsilon$, even if $\boldsymbol{x}_{N+1} \notin \hat{\Xi}$ in (6.42). Assume that $\boldsymbol{x}_i^* \notin \hat{\Xi}$ (although, clearly, $\boldsymbol{x}_i^* \in \Xi$) and $\hat{f}_N \left(\boldsymbol{x}_i^*\right) < \hat{f}_N (\boldsymbol{x_{best}} (N))$ (i.e. the surrogate model of the cost function captures that $\boldsymbol{x}_i^*$ improves upon the current best candidate). Due to how the revisited PSVM classifier works, $\boldsymbol{x_{best}} (N) \in \hat{\Xi}$, see Section 6.1.3 and in particular Problem (6.25). We expect $p_N \left(\boldsymbol{x}_i^* \in \Xi\right)$ to be close to $p_N (\boldsymbol{x_{best}} (N) \in \Xi)$, since the two points are close to each other, although

$$p_N \left(\boldsymbol{x}_i^* \in \Xi\right) < \gamma$$

[2] COBRA [112] initializes all the margins to $\varsigma_{\Xi_{init}} \in \mathbb{R}$. The procedure keeps track of the number of subsequent successful or failed iterations through two counters, $C_{\Xi-feas}, C_{\Xi-infeas} \in \mathbb{N} \cup \{0\}$. Whenever a Ξ-feasible sample is found, $C_{\Xi-infeas}$ is set to zero; vice-versa, when Problem (2.63) returns a Ξ-infeasible point, $C_{\Xi-feas}$ is reset. Once $C_{\Xi-feas} \geq T_{\Xi-feas}$, where $T_{\Xi-feas} \in \mathbb{N}$ is a threshold, COBRA [112] sets $\varsigma_{\Xi}^{(j)} = \varsigma_{\Xi}^{(j)}/2, \forall j = 1, \dots, q_{\Xi}$. Vice-versa, when $C_{\Xi-infeas} \geq T_{\Xi-infeas}$, $T_{\Xi-infeas} \in \mathbb{N}$, the margins are set to $\varsigma_{\Xi}^{(j)} = \max \left\{2 \cdot \varsigma_{\Xi}^{(j)}, \varsigma_{\Xi_{max}}\right\}, \forall j = 1, \dots, q_{\Xi}$, where $\varsigma_{\Xi_{max}} \in \mathbb{R}_{>0}$ is an upper bound on the margins. $\varsigma_{\Xi_{init}}, \varsigma_{\Xi_{max}}, T_{\Xi-feas}$ and $T_{\Xi-infeas}$ are all hyper-parameters of the COBRA [112] algorithm.

under the assumption that $\boldsymbol{x}_i^*$ is classified as Ξ-infeasible. We also consider an additional point,

$$\tilde{\boldsymbol{x}} \in \mathcal{B}\left(\boldsymbol{x}_{\boldsymbol{best}}\left(N\right);\epsilon\right) \cap \Omega, \quad \tilde{\boldsymbol{x}} \notin \hat{\Xi}, \tag{6.44}$$

such that:

$$\begin{aligned}
0 < \quad & p_N\left(\boldsymbol{x}_i^* \in \Xi\right) && \leq && p_N\left(\tilde{\boldsymbol{x}} \in \Xi\right) < \gamma && \leq && p_N\left(\boldsymbol{x}_{\boldsymbol{best}}\left(N\right) \in \Xi\right), && && \text{(6.45a)}\\
0 = \quad & \left\|\boldsymbol{x}_i^* - \boldsymbol{x}_i^*\right\|_2 && \leq && \left\|\tilde{\boldsymbol{x}} - \boldsymbol{x}_i^*\right\|_2 && < && \left\|\boldsymbol{x}_{\boldsymbol{best}}\left(N\right) - \boldsymbol{x}_i^*\right\|_2 && < \quad \epsilon, && \text{(6.45b)}\\
& \hat{f}_N\left(\boldsymbol{x}_i^*\right) && \leq && \hat{f}_N\left(\tilde{\boldsymbol{x}}\right) && < && \hat{f}_N\left(\boldsymbol{x}_{\boldsymbol{best}}\left(N\right)\right), && && \text{(6.45c)}\\
f^* = \quad & f\left(\boldsymbol{x}_i^*\right) && \leq && f\left(\tilde{\boldsymbol{x}}\right) && < && f\left(\boldsymbol{x}_{\boldsymbol{best}}\left(N\right)\right). && && \text{(6.45d)}
\end{aligned}$$

In (6.45), we have assumed that the surrogate model $\hat{f}_N\left(\cdot\right)$ in (4.1) orders $\boldsymbol{x}_i^*, \tilde{\boldsymbol{x}}$ and $\boldsymbol{x}_{\boldsymbol{best}}\left(N\right)$ in the same way as the cost function $f\left(\cdot\right)$ of the GOP (2.1). If $f\left(\cdot\right)$ and $\hat{f}_N\left(\cdot\right)$ are Lipschitz continuous, then (6.45b), (6.45c) and (6.45d) are closely related. Figure 32 depicts the situation described by (6.45).

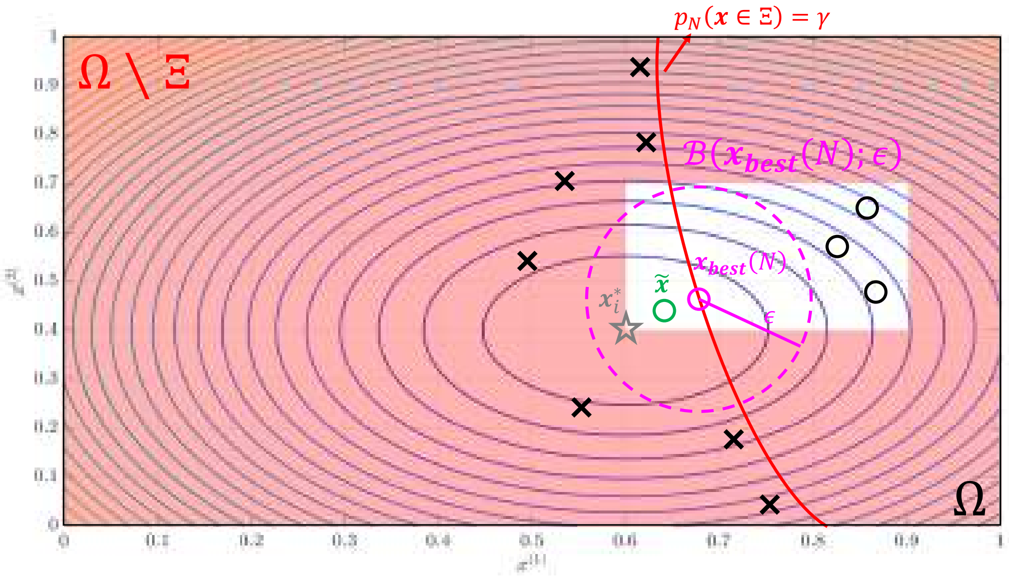


Figure 32: Representation of the situation in (6.45)**. The black circles denote the Ξ-feasible samples available to the optimization procedure whereas the black crosses are the Ξ-infeasible ones. The current best candidate is colored in magenta. Similarly, we depict the open ball $\mathcal{B}\left(\boldsymbol{x}_{\boldsymbol{best}}\left(N\right);\epsilon\right)$ using a dashed line in magenta. The shaded red area depicts the Ξ-infeasible region of the GOP** (2.1) **while the red curve is the decision boundary of the PSVM classifier, trained from the samples at hand. The grey star is a global minimizer of the global optimization problem (which is yet to be found by the BBO or PBO procedure). We also show the level curves of the surrogate model $\hat{f}_N\left(\boldsymbol{x}\right)$ to better grasp** (6.45c)**. Lastly, the sample $\tilde{\boldsymbol{x}} \in \mathcal{B}\left(\boldsymbol{x}_{\boldsymbol{best}}\left(N\right);\epsilon\right) \cap \Omega, \tilde{\boldsymbol{x}} \notin \hat{\Xi}$, is the green circle.**

Now, suppose that we look for the new candidate sample $\boldsymbol{x}_{N+1}$ by solving Problem (6.37) following a pure exploitation approach (i.e. $\delta = 1$). The acquisition function in (6.38) simply becomes:

$$a_N\left(\boldsymbol{x}, \varepsilon_\Xi\right) = \hat{\hat{f}}_N\left(\boldsymbol{x}; \mathcal{X}_{aug}\right) + \varepsilon_\Xi. \tag{6.46}$$

Note that the constraint in (6.40) can be re-written as:

$$\varepsilon_\Xi \geq \frac{\gamma - p_N(\boldsymbol{x} \in \Xi)}{\gamma}. \tag{6.47}$$

Clearly, if $\boldsymbol{x} \in \hat{\Xi}$ (i.e. $p_N(\boldsymbol{x} \in \Xi) \geq \gamma$), then the inequality in (6.47) holds $\forall \varepsilon_\Xi \in [0, 1]$. Vice-versa, if $\boldsymbol{x} \notin \hat{\Xi}$ (i.e. $p_N(\boldsymbol{x} \in \Xi) < \gamma$), then the slack ε_Ξ must be greater than zero in order to satisfy the aforementioned inequality. By substituting (6.47) in (6.46), taking into account that the slack ε_Ξ is constrained to be in the segment $[0, 1]$, we get:

$$a_N(\boldsymbol{x}, \varepsilon_\Xi) \geq \tilde{a}_N(\boldsymbol{x}) = \hat{\bar{f}}_N(\boldsymbol{x}; \mathcal{X}_{aug}) + \max\left\{0, \frac{\gamma - p_N(\boldsymbol{x} \in \Xi)}{\gamma}\right\}. \tag{6.48}$$

Consider $\tilde{\boldsymbol{x}}' \in \Omega$ to be a potential solution of Problem (6.37) with $a_N(\boldsymbol{x}, \varepsilon_\Xi)$ in (6.46). Due to the minimization of the acquisition function, we select the lowest possible value for the slack ε_Ξ, i.e. the expression in (6.48) holds with equality at $\tilde{\boldsymbol{x}}'$:

$$a_N(\tilde{\boldsymbol{x}}', \cdot) = \tilde{a}_N(\tilde{\boldsymbol{x}}') = \hat{\bar{f}}_N(\tilde{\boldsymbol{x}}'; \mathcal{X}_{aug}) + \max\left\{0, \frac{\gamma - p_N(\tilde{\boldsymbol{x}}' \in \Xi)}{\gamma}\right\}. \tag{6.49}$$

Now, let us go back to the situation described by (6.45) and consider the point $\tilde{\boldsymbol{x}}$ in (6.44). Clearly, due to (6.45c), $\tilde{\boldsymbol{x}}$ is also a potential solution of Problem (6.37) when $a_N(\boldsymbol{x}, \varepsilon_\Xi)$ is defined as in (6.46). In particular, $\tilde{\boldsymbol{x}}$ is selected as the new candidate sample (i.e. $\boldsymbol{x}_{N+1} = \tilde{\boldsymbol{x}}$) over the point $\tilde{\boldsymbol{x}}' \in \hat{\Xi}$, even if it is deemed as Ξ-infeasible by the PSVM classifier, whenever:

$$\begin{aligned}
&\tilde{a}_N(\tilde{\boldsymbol{x}}) < \tilde{a}_N(\tilde{\boldsymbol{x}}') \\
&\tilde{\boldsymbol{x}} \notin \hat{\Xi} \implies \varepsilon_\Xi > 0 \qquad \tilde{\boldsymbol{x}}' \in \hat{\Xi} \implies \varepsilon_\Xi = 0 \\
\hat{\bar{f}}_N(\tilde{\boldsymbol{x}}; \mathcal{X}_{aug}) + \frac{\gamma - p_N(\tilde{\boldsymbol{x}} \in \Xi)}{\gamma} &< \hat{\bar{f}}_N(\tilde{\boldsymbol{x}}'; \mathcal{X}_{aug}) \\
\hat{\bar{f}}_N(\tilde{\boldsymbol{x}}'; \mathcal{X}_{aug}) - \hat{\bar{f}}_N(\tilde{\boldsymbol{x}}; \mathcal{X}_{aug}) &> \frac{\gamma - p_N(\tilde{\boldsymbol{x}} \in \Xi)}{\gamma} \\
\frac{\hat{f}_N(\tilde{\boldsymbol{x}}') - \hat{f}_{N_{min}}(\mathcal{X}_{aug})}{\Delta \hat{F}_N(\mathcal{X}_{aug})} - \frac{\hat{f}_N(\tilde{\boldsymbol{x}}) - \hat{f}_{N_{min}}(\mathcal{X}_{aug})}{\Delta \hat{F}_N(\mathcal{X}_{aug})} &> \frac{\gamma - p_N(\tilde{\boldsymbol{x}} \in \Xi)}{\gamma} \\
\hat{f}_N(\tilde{\boldsymbol{x}}') - \hat{f}_N(\tilde{\boldsymbol{x}}) &> \frac{\Delta \hat{F}_N(\mathcal{X}_{aug})}{\gamma} \cdot [\gamma - p_N(\tilde{\boldsymbol{x}} \in \Xi)].
\end{aligned} \tag{6.50}$$

Therefore, if $\tilde{\boldsymbol{x}}$ improves upon $\tilde{\boldsymbol{x}}'$ (for what concerns the surrogate model $\hat{f}_N(\cdot)$ in (4.1)) by more than $\frac{\Delta \hat{F}_N(\mathcal{X}_{aug})}{\gamma} \cdot [\gamma - p_N(\tilde{\boldsymbol{x}} \in \Xi)]$, then Problem (6.37) returns $\boldsymbol{x}_{N+1} = \tilde{\boldsymbol{x}} \notin \hat{\Xi}$. Vice-versa, $\boldsymbol{x}_{N+1} = \tilde{\boldsymbol{x}}' \in \hat{\Xi}$ if (6.50) does not hold. Note that, from (6.45), $\tilde{\boldsymbol{x}}$ could even be the global minimizer $\boldsymbol{x}_i^*$ of the GOP (2.1). Some similar considerations can be made in the case $\delta \neq 1$ for the acquisition function $a_N(\boldsymbol{x}, \varepsilon_\Xi)$ in (6.38).

So far, we have implicitly assumed that Problem (6.37) always admits a solution. In practice, we can prove that, under mild assumptions on the models $\hat{f}_N(\boldsymbol{x})$ in (4.1) and $m_{\Xi_N}(\boldsymbol{x})$ in (6.2), both the acquisition function $a_N(\boldsymbol{x}, \varepsilon_\Xi)$ in (6.38) and the constraint function in (6.40) are differentiable everywhere (and hence continuous). The latter result can then be used to address the existence of a solution for Problem (6.37), as we will see later on in this Section.

***Proposition 6.5: Differentiability of the functions of Problem* (6.37).** *The acquisition function $a_N(\boldsymbol{x}, \varepsilon_\Xi)$ in* (6.38) *is differentiable everywhere with respect to $\begin{bmatrix}\boldsymbol{x} & \varepsilon_\Xi\end{bmatrix}^\top$ if and only if the chosen radial basis function $\phi_{f_i}(\boldsymbol{x}; \epsilon_f) = \varphi_f(\epsilon_f \cdot \|\boldsymbol{x} - \boldsymbol{x}_i\|_2)$ for the surrogate model $\hat{f}_N\left(\boldsymbol{x}; \boldsymbol{\beta}_f, \epsilon_f\right)$ in* (4.1) *is differentiable everywhere.*
Instead, the function associated to the constraint in (6.40)*, namely:*

$$h(\boldsymbol{x}, \varepsilon_\Xi) = \gamma \cdot (1 - \varepsilon_\Xi) - p_N(\boldsymbol{x} \in \Xi), \tag{6.51}$$

is differentiable everywhere with respect to $\begin{bmatrix}\boldsymbol{x} & \varepsilon_\Xi\end{bmatrix}^\top$ if the chosen radial basis function $\phi_{\Xi_i}(\boldsymbol{x}; \epsilon_\Xi) = \varphi_\Xi(\epsilon_\Xi \cdot \|\boldsymbol{x} - \boldsymbol{x}_i\|_2)$ for the model $m_{\Xi_N}(\boldsymbol{x}; \tilde{\boldsymbol{\beta}}_\Xi, \epsilon_\Xi)$ in (6.2) *is differentiable everywhere.*

Proof. Both $a_N(\boldsymbol{x}, \varepsilon_\Xi)$ in (6.38) and $h(\boldsymbol{x}, \varepsilon_\Xi)$ in (6.51) are the sum of multiple terms, some of which depend only on $\boldsymbol{x}$ and others only on ε_Ξ. Therefore, we simply need to assess the differentiability of each addend. Clearly, the terms ε_Ξ in (6.38) and $\gamma \cdot (1 - \varepsilon_\Xi)$ in (6.38) are differentiable everywhere with respect to $\begin{bmatrix}\boldsymbol{x} & \varepsilon_\Xi\end{bmatrix}^\top$ since they are linear in ε_Ξ and do not depend on $\boldsymbol{x}$. Instead, the differentiability of $\hat{\hat{f}}_N(\boldsymbol{x}; \mathcal{X}_{aug})$, $\bar{z}_N(\boldsymbol{x}; \mathcal{X}_{aug})$ in (6.38) and $p_N(\boldsymbol{x} \in \Xi)$ in (6.51) are addressed, respectively, in Propositions 2.1, 4.4 and 6.3. □

***Lemma 6.2: Gradients of the functions of Problem* (6.37).** *Suppose that $a_N(\boldsymbol{x}, \varepsilon_\Xi)$ in* (6.38) *and $h(\boldsymbol{x}, \varepsilon_\Xi)$ in* (6.51) *are differentiable everywhere (see Proposition 6.5). Then, their gradients are:*

$$\begin{aligned}
\nabla_{\boldsymbol{x},\varepsilon_\Xi} a_N(\boldsymbol{x}, \varepsilon_\Xi) &= \begin{bmatrix} \nabla_{\boldsymbol{x}} a_N(\boldsymbol{x}, \varepsilon_\Xi) \\ \frac{\partial}{\partial \varepsilon_\Xi} a_N(\boldsymbol{x}, \varepsilon_\Xi) \end{bmatrix} \\
&= \begin{bmatrix} \frac{\delta}{\Delta \hat{F}_N(\mathcal{X}_{aug})} \cdot \nabla_{\boldsymbol{x}} \hat{f}_N(\boldsymbol{x}) + \frac{1-\delta}{\Delta Z_N(\mathcal{X}_{aug})} \cdot \nabla_{\boldsymbol{x}} z_N(\boldsymbol{x}) \\ 1 \end{bmatrix}, \qquad (6.52a) \\
\nabla_{\boldsymbol{x},\varepsilon_\Xi} h(\boldsymbol{x}, \varepsilon_\Xi) &= \begin{bmatrix} \nabla_{\boldsymbol{x}} h(\boldsymbol{x}, \varepsilon_\Xi) \\ \frac{\partial}{\partial \varepsilon_\Xi} h(\boldsymbol{x}, \varepsilon_\Xi) \end{bmatrix}
\end{aligned}$$

$$= \begin{bmatrix} -\nabla_{\boldsymbol{x}} s_{\Xi_N}(\boldsymbol{x}) \\ -\gamma \end{bmatrix}. \tag{6.52b}$$

In particular, $\nabla_{\boldsymbol{x}} \hat{f}_N(\boldsymbol{x}), \nabla_{\boldsymbol{x}} z_N(\boldsymbol{x})$ *and* $\nabla_{\boldsymbol{x}} s_{\Xi_N}(\boldsymbol{x})$ *are defined respectively as in* (2.22), (5.1) *and* (6.22).

Proof. The Proof is straightforward due to the fact that the addends of $a_N(\boldsymbol{x}, \varepsilon_\Xi)$ in (6.38) and $h(\boldsymbol{x}, \varepsilon_\Xi)$ in (6.51) either depend on $\boldsymbol{x}$ or on ε_Ξ. The differentiation of the two functions with respect to $\boldsymbol{x}$ yields the same results reported in Lemma 5.2 and in Lemma 6.1. Instead, the partial derivative of $a_N(\boldsymbol{x}, \varepsilon_\Xi)$ in (6.38) and $h(\boldsymbol{x}, \varepsilon_\Xi)$ in (6.51) with respect to ε_Ξ are easy to to compute since both functions are linear in the slack. □

We now go back to addressing the existence of a solution for Problem (6.37)

Proposition 6.6. *Suppose that the completely known constraint set* Ω *of the GOP* (2.1) *is compact. Then, provided that the conditions in Proposition 6.5 are verified, Problem* (6.37) *always admits a solution.*

Proof. The constraint set of Problem (6.37) is the intersection of two sets,

$$\left\{ \begin{bmatrix} \boldsymbol{x} & \varepsilon_\Xi \end{bmatrix}^\top : \boldsymbol{x} \in \Omega, \varepsilon_\Xi \in [0, 1] \right\} \tag{6.53}$$

and

$$\left\{ \begin{bmatrix} \boldsymbol{x} & \varepsilon_\Xi \end{bmatrix}^\top : \boldsymbol{x} \in \mathbb{R}^n, \varepsilon_\Xi \in \mathbb{R}, \gamma \cdot (1 - \varepsilon_\Xi) - p_N(\boldsymbol{x} \in \Xi) \leq 0 \right\}, \tag{6.54}$$

resulting in:

$$\begin{aligned} \Big\{ \begin{bmatrix} \boldsymbol{x} & \varepsilon_\Xi \end{bmatrix}^\top : \boldsymbol{x} \in \Omega, \varepsilon_\Xi \in [0, 1], \\ \gamma \cdot (1 - \varepsilon_\Xi) - p_N(\boldsymbol{x} \in \Xi) \leq 0 \Big\}. \end{aligned} \tag{6.55}$$

Clearly, if Ω is compact, then so is the set in (6.53). The set in (6.54) is also compact since the constraint function in (6.51) is differentiable everywhere, due to Proposition 6.5, and hence continuous [114]. Thus, the constraint set in (6.55) is compact being the intersection of two compact sets (see Proposition A.5 in Appendix A.2). Finally, by Proposition 6.5, the acquisition function $a_N(\boldsymbol{x}, \varepsilon_\Xi)$ in (6.38) is continuous and thus Problem (6.37) admits a solution due to the Extreme Value Theorem 1.1. □

We conclude this Section by pointing out that, as an alternative to the slack formulation in Problem (6.37), we could look for new candidate samples by solving:

$$\begin{aligned} \boldsymbol{x}_{N+1} = \arg\min_{\boldsymbol{x}} & \; a_N(\boldsymbol{x}) \\ \text{s.t.} \quad & \boldsymbol{x} \in \Omega \\ & p_N(\boldsymbol{x} \in \Xi) \geq \gamma, \end{aligned} \tag{6.56}$$

where $a_N(\boldsymbol{x})$ is defined as in (5.12). Then, we could cycle the threshold γ (which defines the decision boundary of the PSVM classifier, see (6.20)) in a similar fashion to the greedy δ-cycling strategy in Section 5.2.3. To do so, analogously to (5.15), we would need to define a cycling set:

$$\Gamma_{cycle} = \langle \gamma_0, \ldots, \gamma_{\tilde{N}_{cycle-1}} \rangle, \quad \tilde{N}_{cycle} \in \mathbb{N}, \gamma_j \in [0, 1], \forall \gamma_j \in \Gamma_{cycle},$$

and keep γ of Problem (6.56) unaltered as long as there is an improvement (see Algorithm 9). However, we claim that the proposed Problem (6.37) better captures the exploration-exploitation-penalization rationale. At the same time, *Problem* (6.56) *can be useful for high-dimensional GOPs* (2.1) *and/or in the case we want to avoid sampling outside* Ξ (e.g. if the sample evaluations involve real-world experiments and Ξ-infeasible tunings might damage the equipment). For example, we could simply set $\Gamma_{cycle} = \langle \gamma_0 \rangle$ with $\gamma_0 \gg 0.5$ (no cycling) to obtain a more conservative constrained black-box or preference-based optimization procedure. Furthermore, for high-dimensional constrained BBO or PBO problems, a more realistic goal is to find good local solutions of the GOP (2.1), starting from at least one Ξ-feasible sample (see Remark 2.1), favoring Problem (6.56) over Problem (6.37).

6.2.1 A note on greedy δ-cycling

In the constrained BBO and PBO frameworks, compared to the greedy δ-cycling approach in the unconstrained case (Section 5.2.3), new candidate samples $\boldsymbol{x}_{N+1}$ might not improve upon the current best candidate $\boldsymbol{x_{best}}(N)$ even if they achieve lower costs or are strictly preferred to the latter. Consequently, we must update the cycling rule in (5.16) to make it consistent with the definition of improvement in Algorithm 9. Suppose that, at iteration k, we have at our disposal $|\mathcal{X}| = N$ samples and denote the trade-off parameter δ in (6.38) as $\delta(k)$ to highlight the iteration number. Furthermore, assume $\delta(k) = \delta_j \in \Delta_{cycle}$ in (5.15), which has been used to find the new candidate sample $\boldsymbol{x}_{N+1}$ at iteration k by solving Problem (6.37). Then, at iteration $k+1$, we select $\delta(k+1) \in \Delta_{cycle}$ as:

$$\delta(k+1) = \begin{cases} \delta(k) & \text{if } \boldsymbol{x_{best}}(N), \boldsymbol{x}_{N+1} \in \Xi \text{ and} \\ & y_{N+1} < y_{best}(N) \text{ (BBO) or } \boldsymbol{x}_{N+1} \succ \boldsymbol{x_{best}}(N) \text{ (PBO)} \\ \delta_{(j+1) \bmod N_{cycle}} & \text{otherwise} \end{cases} . \tag{6.57}$$

Note that we do not need to explicitly address the case $\boldsymbol{x_{best}}(N) \notin \Xi$. That is because, whenever the current best candidate is Ξ-infeasible, then so are all the samples in $\mathcal{X}$; therefore, new candidate samples are sought by solving Problem (6.34) instead of Problem (6.37). The greedy δ-cycling in (6.57) starts as soon as $\emptyset \subset \mathcal{X} \cap \Xi \subseteq \mathcal{X}$. Moreover, any Ξ-infeasible point does not improve upon $\boldsymbol{x_{best}}(N) \in \Xi$, even if it achieves a lower cost or is strictly preferred to the current best candidate, resulting in $\delta(k+1) = \delta_{(j+1)\mathrm{mod}N_{cycle}}$. Lastly, whenever $\boldsymbol{x_{best}}(N), \boldsymbol{x}_{N+1} \in \Xi$, the cycling rule in (6.57) is equivalent to the one in (5.16). Hence, the same considerations also hold for Problem (6.36) with $a_N(\boldsymbol{x})$ in (5.12).

6.3 Algorithms

Algorithm 16 describes each step of the C-GLIS-r and C-GLISp-r procedures in detail. Similarly to what we did in Section 5.3, where we formalized algorithms GLIS-r [108] and GLISp-r [109], we analyze BBO and PBO jointly. Furthermore, due to how we estimate the probability of Ξ-feasibility (i.e. through a PSVM classifier), Algorithm 16 is able to handle both measurable black-box constraints functions and decision-maker-based constraints. In the former case, i.e. when the set $\mathcal{C}_\Xi$ in (2.12) is available, we can easily obtain $\mathcal{U}_\Xi$ in (2.11) from $\mathcal{C}_\Xi$. Similarly to Algorithm 14, we propose to perform the recalibration of the hyper-parameters of the surrogate model $\hat{u}_{\Xi_N}(\boldsymbol{x})$ in (6.19) (Section 6.1.4) only at certain iterations of the C-GLIS-r and C-GLISp-r procedures, as highlighted by the set $\mathcal{K}_{R_\Xi} \subseteq \{1, \ldots, N_{max} - N_{init}\}$.

C-GLIS (respectively, C-GLISp [156], in Algorithm 12) and C-GLIS-r (C-GLISp-r, in Algorithm 16) differ from each other on several aspects. Other than the modifications introduced by GLIS-r [108] and GLISp-r [109] (see Algorithm 14), the main differences between the latter methods are: (i) the black-box constraint set Ξ of the GOP (2.1) is handled by different surrogates, (ii) C-GLIS-r/C-GLISp-r adopt an infill sampling criterion that depends on how many Ξ-feasible samples are present in $\mathcal{X}$ (2.9) while C-GLIS/C-GLISp [156] do not and, consequently, (iii) the methods rely on different acquisition functions/infill sampling criteria. Furthermore, we point out that we still use the original IDW distance function $z_N(\boldsymbol{x})$ in (4.17) instead of the revisited one in (4.18). In any case, in Algorithm 16, we explicit in grey the differences between C-GLIS/C-GLISp [156] and C-GLIS-r/C-GLISp-r.

As final remark, note that Algorithm 16 (C-GLIS-r and C-GLISp-r) is equivalent to Algorithm 14 (GLIS-r [108] and GLISp-r [109]) whenever no black-box constraints are present (i.e. when $\Xi = \mathbb{R}^n$ for the GOP (2.1)).

Algorithm 16: C-GLIS-r and C-GLISp-r

Input: (i) A-priori known constraint set Ω of the GOP (2.1); (ii) Number of initial samples $N_{init} \in \mathbb{N}$; (iii) Budget $N_{max} \in \mathbb{N}, N_{max} > N_{init}$; (iv) Hyper-parameters for the surrogate model $\hat{f}_N(\boldsymbol{x})$ in (4.1): shape parameter $\epsilon_f \in \mathbb{R}_{>0}$ and radial function $\varphi_f(\cdot)$ (BBO and PBO), threshold $\epsilon_{SVD} \in \mathbb{R}_{>0}$ (BBO), regularization parameter $\lambda_f \in \mathbb{R}_{\geq 0}$ and tolerance $\sigma_\pi \in \mathbb{R}_{>0}$ (PBO); (v) *Hyper-parameters for the surrogate model $\hat{u}_{\Xi_N}(\boldsymbol{x})$ in* (6.19)*, namely shape parameter $\epsilon_\Xi \in \mathbb{R}_{>0}$, radial function $\varphi_\Xi(\cdot)$, trade-off weight $C_{SVM} \in \mathbb{R}_{>0}$ and threshold $\gamma \in [0,1]$*; (vi) *Cycling set Δ_{cycle} in* (5.15) *for the infill sampling criterion*; (vii) *Number of clusters $K_{aug} \in \mathbb{N}$ for the augmented sample set $\mathcal{X}_{aug}$ generated by Algorithm 13*; (viii) Set of models $\mathcal{M}_f$ in (4.30) for the recalibration of the surrogate model $\hat{f}_N(\boldsymbol{x})$ in (4.1); (ix) Set of indexes for the recalibration of the surrogate model $\hat{f}_N(\boldsymbol{x})$ in (4.1), $\mathcal{K}_{R_f} \subseteq \{1, \ldots, N_{max} - N_{init}\}$, and folds ratio $R_f \in [0,1]$; (x) *Bounds on the hyper-parameters of the PSVM classifier $\boldsymbol{l}_\Xi, \boldsymbol{u}_\Xi \in \mathbb{R}^2_{>0}$*; (xi) *Set of indexes for the recalibration of the surrogate model $\hat{u}_{\Xi_N}(\boldsymbol{x})$ in* (6.19)*, $\mathcal{K}_{R_\Xi} \subseteq \{1, \ldots, N_{max} - N_{init}\}$, and folds ratio $R_\Xi \in [0,1]$.*

Output: (i) Best cost obtained by the procedure $y_{best}(N_{max})$ (only for BBO); (ii) Best sample obtained by the procedure $\boldsymbol{x_{best}}(N_{max})$.

```
1: Rescale the GOP (2.1) as proposed in Section 4.4.1, obtaining Problem (4.27)
2: Generate a set $\mathcal{X}$ in (2.9) of $N_{init}$ starting points using a LHD (see Section 2.4)
3: Evaluate the samples in $\mathcal{X}$ either by measuring the values of $f(\cdot)$, obtaining the set $\mathcal{Y}$ in (2.10) (BBO), or
   by querying the decision-maker as in Algorithm 7, obtaining the sets $\mathcal{B}$ (3.9) and $\mathcal{S}$ (3.10), as well as the
   best candidate $\boldsymbol{x_{best}}(N_{init})$
4: Evaluate the $\Xi$-feasibility of each $\boldsymbol{x}_i \in \mathcal{X}$, obtaining the set $\mathcal{U}_\Xi$ in (2.11)
5: Set $N = N_{init}$ (and $M = |\mathcal{B}|$ for PBO)
6: Set $\delta = \delta_0 \in \Delta_{cycle}$ and $j = 0$
7: for $k = 1, 2, \ldots, N_{max} - N_{init}$ do
8:     if $k \in \mathcal{K}_{R_f}$ then recalibrate the surrogate model $\hat{f}_N(\boldsymbol{x})$ in (4.1) as in Algorithm 8
9:     if $k \in \mathcal{K}_{R_\Xi}$ then recalibrate the surrogate model $\hat{u}_{\Xi_N}(\boldsymbol{x})$ in (6.19) as in Section 6.1.4
10:    if $\exists u_i \in \mathcal{U}_\Xi$ such that $u_i = 1$ then build the surrogate model for $f(\boldsymbol{x})$ using the information at hand: in
       BBO, find $\boldsymbol{\beta}_f$ as in (4.4) whereas, for PBO, select $\boldsymbol{\beta}_f$ as the solution of Problem (4.10)
11:    if $\exists u_i, u_j \in \mathcal{U}_\Xi$ such that $u_i = 1, u_j = 0, i \neq j$ then estimate the probability of $\Xi$-feasibility $p_N(\boldsymbol{x} \in \Xi)$
       in (6.15) as proposed in Section 6.1.3
12:    Generate the augmented sample set $\mathcal{X}_{aug}$ through Algorithm 13
13:    Look for the next candidate sample $\boldsymbol{x}_{N+1}$:
       a: if $u_i = 0, \forall u_i \in \mathcal{U}_\Xi$ then solve Problem (6.34) with $z_N(\boldsymbol{x})$ in (4.17)
       b: if $u_i = 1, \forall u_i \in \mathcal{U}_\Xi$ then solve Problem (6.36) with $a_N(\boldsymbol{x})$ in (5.12)
       c: if $\exists u_i, u_j \in \mathcal{U}_\Xi$ such that $u_i = 1, u_j = 0, i \neq j$ then solve Problem (6.37) with $a_N(\boldsymbol{x}, \varepsilon_\Xi)$ in (6.38)
          and $p_N(\boldsymbol{x} \in \Xi)$ in (6.15)
14:    Evaluate the new candidate sample, obtaining either $y_{N+1} = f(\boldsymbol{x}_{N+1})$ (in BBO) or $b_{M+1} =$
       $\pi_{\gtrsim}(\boldsymbol{x}_{N+1}, \boldsymbol{x_{best}}(N))$ (in PBO)
15:    Evaluate the $\Xi$-feasibility of $\boldsymbol{x}_{N+1}$, obtaining $u_{N+1} = u_\Xi(\boldsymbol{x}_{N+1})$
16:    Update the set of samples $\mathcal{X}$ and either the set of measures of $f(\cdot)$, $\mathcal{Y}$ (BBO), or the preference
       information in the sets $\mathcal{B}$ and $\mathcal{S}$ (PBO)
17:    Update the $\Xi$-feasibility information in the set $\mathcal{U}_\Xi$
18:    Check if $\boldsymbol{x}_{N+1}$ improves upon $\boldsymbol{x_{best}}(N)$, as highlighted by the flag $hasImproved$ from Algorithm 9
19:    if $hasImproved$ then
20:        Set $\boldsymbol{x_{best}}(N+1) = \boldsymbol{x}_{N+1}$
21:    else
22:        Keep $\boldsymbol{x_{best}}(N+1) = \boldsymbol{x_{best}}(N)$
23:        if $\exists u_i \in \mathcal{U}_\Xi$ such that $u_i = 1$ then set $\delta = \delta_{(j+1) mod N_{cycle}} \in \Delta_{cycle}$ (greedy $\delta$-cycling) and $j = j + 1$
24:    Set $N = N + 1$ (and $M = M + 1$ for PBO)
```

6.3.1 A note on the convergence

Algorithm 16 does not guarantee its convergence to a global solution of the GOP (2.1) *if* $\Xi \neq \mathbb{R}^n$ (i.e.

whenever we are actually dealing with constrained BBO or PBO). As long as no Ξ-feasible samples are available, `C-GLIS-r` and `C-GLISp-r` look for new candidate samples $\boldsymbol{x}_{N+1} \in \Omega$ by solving Problem (6.34). In Section 5.3.1, we have seen that, if Ω is compact, then selecting $\boldsymbol{x}_{N+1}$ through the minimization of the IDW distance function $z_N(\boldsymbol{x})$ in (4.17) over Ω makes the set of samples $\mathcal{X}$ in (2.9) progressively more space-filling. Moreover, if Problem (6.34) (pure exploration) were to be solved infinitely often, then the sequence of iterates generated by Algorithm 16 would be dense in Ω. However, in this case, we do not guarantee that Problem (6.34) is solved infinitely often. In fact, as soon as a Ξ-feasible sample is found, we look for the minimizers of Problem (6.37) instead of those of Problem (6.34). Even if $\delta = 0$ for the acquisition function $a_N(\boldsymbol{x}, \varepsilon_\Xi)$ in (6.38), the exploration of Ω is penalized by the constraint in (6.40). For example, let us go back to the situation in (6.45) but this time consider $\delta = 0$ instead of $\delta = 1$. Similarly to Section 6.2, let $\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{x}}' \in \Omega$ be two potential solutions of Problem (6.37) such that $\tilde{\boldsymbol{x}} \notin \hat{\Xi}$ and $\tilde{\boldsymbol{x}}' \in \hat{\Xi}$, with $\hat{\Xi}$ defined as in (6.42). Then $\tilde{\boldsymbol{x}}$ is selected over $\tilde{\boldsymbol{x}}'$ if (see the derivation of (6.50)):

$$z_N(\tilde{\boldsymbol{x}}') > \frac{\Delta Z_N\left(\mathcal{X}_{aug}\right)}{\gamma} \cdot \left[\gamma - p_N(\tilde{\boldsymbol{x}} \in \Xi)\right] + z_N(\tilde{\boldsymbol{x}}). \tag{6.58}$$

Therefore, in this case, Algorithm 16 might select new candidate samples that are deemed as Ξ-infeasible by the PSVM classifier if they provide good exploratory capabilities (low $z_N(\tilde{\boldsymbol{x}})$) but, at the same time, do not violate the constraint in (6.39) by too much $\left(\frac{\Delta Z_N(\mathcal{X}_{aug})}{\gamma} \cdot \left[\gamma - p_N(\tilde{\boldsymbol{x}} \in \Xi)\right] \text{ small}\right)$.

The easiest way to ensure the asymptotic global convergence of `C-GLIS-r` *and* `C-GLISp-r` *is to perform pure unpenalized exploration infinitely often.* For example, whenever $\delta = 0$ for the acquisition function $a_N(\boldsymbol{x}, \varepsilon_\Xi)$ in (6.38), we could solve:

$$\begin{aligned} \boldsymbol{x}_{N+1} = \arg\min_{\boldsymbol{x}} & \; z_N(\boldsymbol{x}) \\ \text{s.t.} & \quad \boldsymbol{x} \in \Omega \end{aligned} \tag{6.59}$$

instead of Problem (6.37). Assume that both Ξ and Ω are compact, then the constraint set of the GOP (2.1), namely $\Omega \cap \Xi$, is also compact (and nonempty due to Assumption 2.2). If the conditions in

Theorem 5.2 are verified[3], then, by solving Problem (6.59) infinitely often, the optimization procedures produce sequences of iterates $\langle \boldsymbol{x}_i \rangle_{i\geq 1}$ that are dense in Ω (this can be demonstrated using the same arguments in the Proof of Theorem 5.2). Consequently, $\langle \boldsymbol{x}_i \rangle_{i\geq 1}$ must also be dense in $\Omega \cap \Xi$. Hence, the `C-GLIS-r` and `C-GLISp-r` algorithms which perform pure unpenalized exploration (when $\delta = 0$) converge to the global minimum of the GOP (2.1). The next Example compares the performances achieved by Algorithm 16 against its variant equipped with the just described pure unpenalized exploration approach.

Example 6.4: Pure unpenalized exploration

Consider the `sasena 2` [125] benchmark optimization problem in Appendix B, which has one global minimizer $\boldsymbol{x}^* = \begin{bmatrix} 0.2017 & 0.8332 \end{bmatrix}^\top$ and one local minimizer $\boldsymbol{x}^+ = \begin{bmatrix} 0.2616 & 0.1216 \end{bmatrix}^\top$ ($f(\boldsymbol{x}^*) < f(\boldsymbol{x}^+)$). Its black-box constraint set Ξ defines two disconnected regions, one contains $\boldsymbol{x}^*$ and the other $\boldsymbol{x}^+$. We perform constrained black-box optimization using Algorithm 16 (`C-GLIS-r`) with the following hyper-parameters:

- Number of initial samples $N_{init} = 12$,
- Budget $N_{max} = 100$,
- Hyper-parameters for the surrogate model $\hat{f}_N(\boldsymbol{x})$ in (4.1): $\epsilon_f = 0.5378$, $\varphi_f(\cdot)$ inverse quadratic, $\epsilon_{SVD} = 10^{-6}$,
- Hyper-parameters for the surrogate model $\hat{u}_{\Xi_N}(\boldsymbol{x})$ in (6.19): $\epsilon_\Xi = 1$, $\varphi_\Xi(\cdot)$ Gaussian, $C_{SVM} = 10^6$, $\gamma = 0.5$,
- Cycling set $\Delta_{cycle} = \langle 0.95, 0.7, 0.35, 0 \rangle$,
- Number of clusters for $\mathcal{X}_{aug}$: $K_{aug} = 5$,
- No recalibration of $\hat{f}_N(\boldsymbol{x})$ in (4.1), i.e. $\mathcal{K}_{R_f} = \emptyset$, $\mathcal{M}_f = \emptyset$,
- No recalibration of $\hat{u}_{\Xi_N}(\boldsymbol{x})$ in (6.19), i.e. $\mathcal{K}_{R_\Xi} = \emptyset$.

We consider two different formulations of the `C-GLIS-r` algorithm. One is the original formulation of the procedure, the other performs pure unpenalized exploration (Problem (6.59)) whenever $\delta = 0$ for the acquisition function $a_N(\boldsymbol{x}, \varepsilon_\Xi)$ in (6.38). We refer to the former as *safe* while the latter as *unpenalized*. Both safe-`C-GLIS-r` and unpenalized-`C-GLIS-r` are started

[3]Formally, we should also ensure that the surrogate model $\hat{f}_N(\boldsymbol{x})$ in (4.1) and the probability of Ξ-feasibility $p_N(\boldsymbol{x} \in \Xi)$ in (6.15) are continuous. However, that is always the case when $\varphi_f(\cdot)$ and $\varphi_\Xi(\cdot)$ for the latter functions are selected as any of the radial functions reported in Definition 2.6.

from the same $\mathcal{X}_{init_1}$, $\left|\mathcal{X}_{init_1}\right| = N_{init}$, samples. For the safe formulation, we also consider an additional set of initial samples $\mathcal{X}_{init_2}$, $\left|\mathcal{X}_{init_2}\right| = N_{init}$, generated by a LHD (Section 2.4) with a different random seed. In any case, the optimization problems related to the infill sampling criteria of the methods (Section 6.2) are solved by the `PSWARM` [72] algorithm (see Section 1.2.5).

Figure 33 depicts the results. Notice how $\hat{\Xi}$ in (6.42) of unpenalized-`C-GLIS-r` is able to roughly capture both disconnected regions defined by Ξ of the `sasena 2` [125] benchmark optimization problem. That is due to the fact that the method explores Ω more. For this reason, it returns a point $\boldsymbol{x_{best}}\left(N_{max}\right)$ that is close to $\boldsymbol{x}^*$. Instead, safe-`C-GLIS-r` gets stuck on the local minimizer $\boldsymbol{x}^+$, when started from $\mathcal{X}_{init_1}$. However, when the latter procedure begins from $\mathcal{X}_{init_2}$ (which contains a point located in the region of Ξ where the global minimizer is located), it favors sampling in a neighborhood of $\boldsymbol{x}^*$. In any case, at least in this Example, $\hat{\Xi}$ in (6.42) estimated by safe-`C-GLIS-r` only captures one of the two disconnected regions of Ξ of the GOP (2.1). *The global convergence of unpenalized-`C-GLIS-r` comes at a cost: several more Ξ-infeasible points are evaluated compared to safe-`C-GLIS-r`.*

Although global convergence is a desirable property for any BBO or PBO algorithm, when black-box constraints are present it is often better to minimize the number of Ξ-infeasible samples tried by the optimization procedure. In particular, as n increases, exploration becomes exponentially harder and hence it is best to limit our search only within a region of Ω where it is more likely to find Ξ-feasible samples (see Remark 2.1). In practice, we would rather find a suboptimal Ξ-feasible calibration (e.g. a local minimizer of the GOP (2.1)) instead of wasting many sample evaluations surveying Ω in the hope of finding a better promising Ξ-feasible region.

6.3.2 A note on the generalization

Algorithm 16 can easily be generalized to handle any surrogate model of the cost function $f(\boldsymbol{x})$ of the GOP (2.1), any exploration function $z_N(\boldsymbol{x})$ and any adequate classifier that returns the probability of Ξ-feasibility $p_N(\boldsymbol{x} \in \Xi) = p\left(\boldsymbol{x} \in \Xi \,\middle|\, —, \boldsymbol{x}\right)$[4]. Following the same rationale of `gMRS` [108] (Section 5.4, Algorithm 15), we only require $\hat{f}_N(\boldsymbol{x})$ and $p_N(\boldsymbol{x} \in \Xi)$ to be continuous, while $z_N(\boldsymbol{x})$ must be proper (see Definition 5.4). The latter conditions ensure that Problems (6.34), (6.36) and (6.37) always admit a solution. For example, in Chapter 7, we will test the `C-GLIS-r` and `C-GLISp-r` procedures

[4]In practice, the probability of Ξ-feasibility could be estimated from $\mathcal{X}$ in (2.9) and $\mathcal{C}_\Xi$ in (2.12) instead of $\mathcal{X}$ and $\mathcal{U}_\Xi$ in (2.11), hence the — in the above expression.

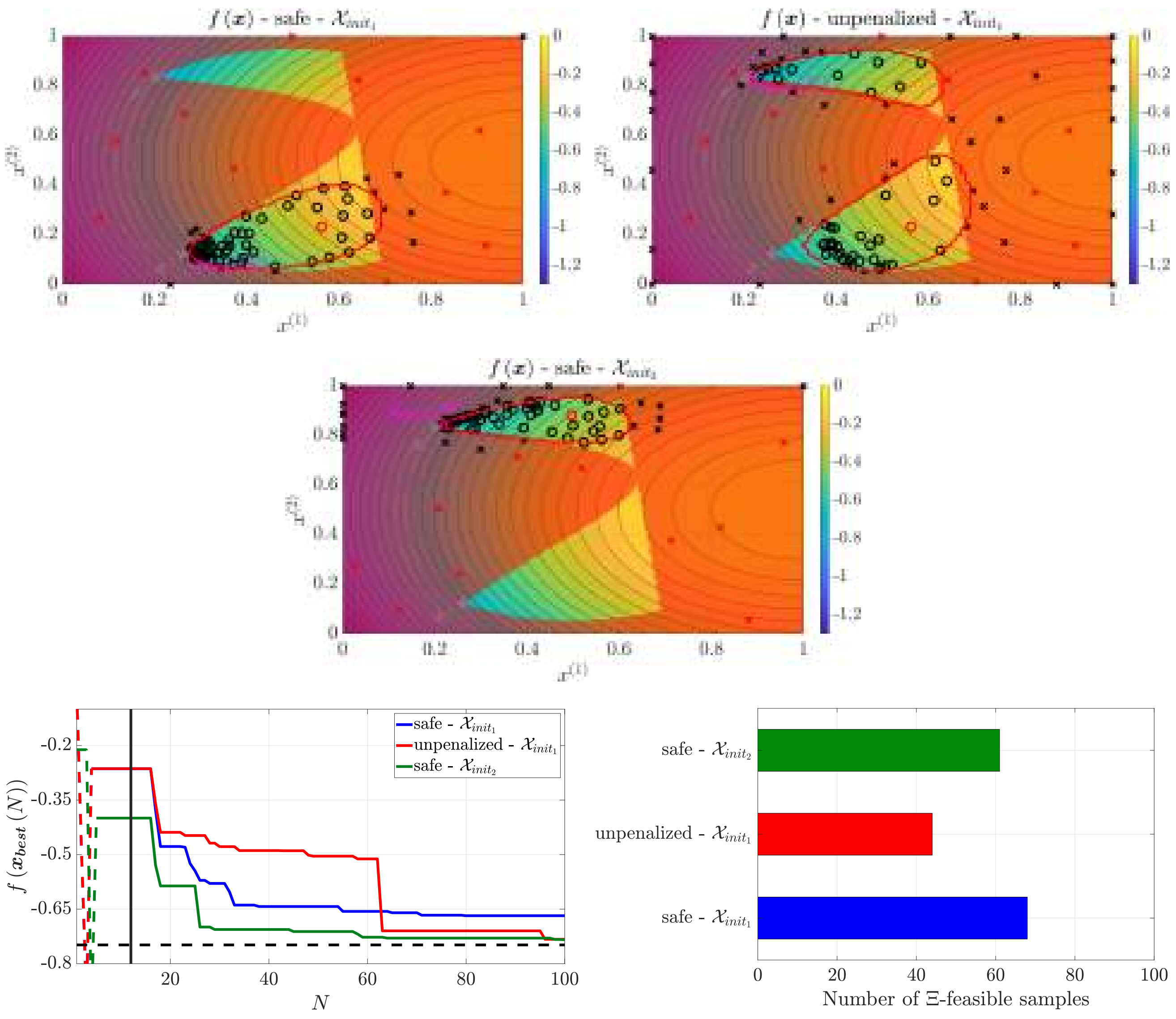


Figure 33: Comparison of the performances achieved by the two formulations of C-GLIS-r described in Example 6.4. The top and middle rows display the samples obtained by C-GLIS-r. The shaded red area denotes the Ξ-infeasible region. The Ξ-feasible samples are depicted with circles whereas the Ξ-infeasible ones are the crosses. We distinguish the N_{init} initial points (in red) from those obtained by the infill sampling criteria (in black). The best candidates are colored in magenta. The stars in grey are the global and local minimizers of the sasena 2 [125] GOP (2.1). Lastly, the red lines are the decision boundaries of the PSVM classifiers. On the bottom left: performances achieved by the different formulations of C-GLIS-r. We depict $f(\boldsymbol{x}_{best}(N))$ as a dashed line as long as $\boldsymbol{x}_{best}(N)$ is Ξ-infeasible and switch to a continuous line when the latter becomes Ξ-feasible. The dashed black-line is the global minimum of the GOP (2.1). The black vertical line denotes N_{init}. On the bottom right: number of Ξ-feasible samples when $N = N_{max}$ for the different formulations of C-GLIS-r.

with $p_N(\boldsymbol{x} \in \Xi)$ estimated through inverse distance weighting interpolation, i.e. as in (4.16). Note that $p_N(\boldsymbol{x} \in \Xi)$ in (4.16) is differentiable everywhere (see Proposition 4.3), making it continuous and thus suitable for the task at hand.

6.4 Chapter summary

In this Chapter, we have presented the fourth contribution of this book, which is the extension of the `GLIS-r` [108] and `GLISp-r` [109] procedures to the constrained black-box and preference-based optimization frameworks, giving rise to `C-GLIS-r` and `C-GLISp-r` respectively. The latter methods estimate the probability of Ξ-feasibility, $p_N(\boldsymbol{x} \in \Xi)$, through a probabilistic support vector machine classifier and use it to penalize the search in those regions of Ω of the GOP (2.1) that are likely to contain Ξ-infeasible samples. We have designed a PSVM classifier that is specifically tailored for constrained BBO and PBO. In particular, we guarantee that, whenever $\boldsymbol{x_{best}}(N)$ is Ξ-feasible, then it is classified as such by the proposed PSVM classifier. `C-GLIS-r` and `C-GLISp-r` employ an infill sampling criterion that behaves differently based on how many Ξ-feasible are present in $\mathcal{X}$ (2.9): (i) if none are available, then we explore Ω in the hope of finding a Ξ-feasible region; (ii) if $\boldsymbol{x}_i \in \Xi, \forall \boldsymbol{x}_i \in \mathcal{X}$, then we adopt the same infill sampling criterion of `GLIS-r` [108] and `GLISp-r` [109]; (iii) if $\exists \boldsymbol{x}_i, \boldsymbol{x}_j \in \mathcal{X}, \boldsymbol{x}_i \neq \boldsymbol{x}_j$, such that $\boldsymbol{x}_i \in \Xi$ and $\boldsymbol{x}_j \notin \Xi$, then `C-GLIS-r` and `C-GLISp-r` look for new candidate samples by trading off exploration and exploitation but, at the same time, penalize the search in those regions of Ω that are likely to contain Ξ-infeasible samples (as predicted by $p_N(\boldsymbol{x} \in \Xi)$). The main difference between the proposed infill sampling criterion and the strategies used by the most popular constrained BBO methods is the addition of a slack variable, which allows the violation of the constraint $p_N(\boldsymbol{x} \in \Xi) \geq \gamma$ when there exist very promising calibrations on the other side of the decision boundary of the PSVM classifier.

Formally, `C-GLIS-r` and `C-GLISp-r` are not globally convergent by design, although they can easily be made so by letting the procedures perform pure unpenalized exploration (whenever needed). In practice, the latter approach is not recommended since it can lead to trying more Ξ-infeasible samples than we are willing to evaluate.

`C-GLIS-r` and `C-GLISp-r` will be compared to `C-GLIS` and `C-GLISp` [156] in Chapter 7. Therein, we will show that the proposed methods can be more sample efficient than the original methods, both in constrained BBO and PBO.

Part III

Empirical results and case study

Chapter 7. Empirical results

In this Chapter, we analyze the performances of the proposed algorithms, namely `GLIS-r` [108], `GLISp-r` [109] (Chapter 5, Algorithm 14), `C-GLIS-r` and `C-GLISp-r` (Chapter 6, Algorithm 16), on several benchmark optimization problems, reported in Appendix B. We compare the proposed procedures against the original methods: `GLIS` [10], `GLISp` [11], `C-GLIS` and `C-GLISp` [156] (Chapter 4, Algorithms 10, 11 and 12).

In the unconstrained BBO and PBO frameworks, we compare `GLIS-r` [108] (respectively, `GLISp-r` [109]) to both `GLIS` [10] (`GLISp` [11]) and `C-GLIS` (`C-GLISp` [156]), even if no black-box constraints are present. That is because `C-GLIS` and `C-GLISp` [156] use the IDW distance function $z_N(\boldsymbol{x})$ in (4.18) instead of the one in (4.17) which, as claimed in [156], is better suited for escaping local minima of the GOP (2.1).

In the constrained BBO and PBO frameworks, we test both the PSVM classifier described in Section 6.1.3 and the IDWI function in (4.16) as estimators of the probability of Ξ-feasibility for `C-GLIS-r` and `C-GLISp-r`. Recall that both `C-GLIS` and `C-GLISp` [156] use the IDWI function in (4.16) to penalize the search in those regions of Ω which are likely to contain Ξ-infeasible samples, see the acquisition functions in (4.20c) and (4.20d); hence, it is interesting to compare the methods when they all use the same surrogates but different infill sampling criteria.

The rest of this Chapter is organized as follows. Section 7.1 describes the experimental setup used for all benchmark optimization problems. In particular, we address the selection of the hyper-parameters for each method and the criteria used when comparing the optimization procedures. Section 7.2 reports the results achieved by `GLIS` [10], `GLISp` [11], `C-GLIS`, `C-GLISp` [156], `GLIS-r` [108] and `GLISp-r` [109] in the unconstrained BBO and PBO frameworks. Instead, Section 7.3 shows the performances of `C-GLIS`, `C-GLISp` [156], `C-GLIS-r` and `C-GLISp-r` in the constrained BBO and PBO frameworks. Lastly, Section 7.4 gives some concluding remarks.

7.1 Experimental setup

All benchmark optimization problems have been solved on a machine with two Intel Xeon E5-2687W @3.00GHz CPUs and 128GB of RAM. `GLIS-r` [108], `GLISp-r` [109], `C-GLIS-r` and `C-GLISp-r` have been implemented in MATLAB. Similarly, we have used the MATLAB code for `GLIS` [10] and `GLISp` [11] provided by the authors (formally, version 2.4 of the software package) and the MATLAB code for `C-GLIS` and `C-GLISp` [156] supplied in version 3.0 of the same software package[1]. All the global optimization problems associated to the infill sampling criteria of the procedures have been solved using the `PSWARM` [72] algorithm (see Section 1.2.5) and with the same settings. In particular, we have used the MATLAB implementation of `PSWARM` [72] provided by [80, 146, 147][2]. Whenever completely known constraints are present (for example, in Problem (6.37)), we have equipped the `PSWARM` [72] procedure with a quadratic penalty function as described in Section 1.2.6.

7.1.1 Hyper-parameters for the procedures

One of the main difficulties when comparing (fairly) surrogate-based methods is the choice of their respective hyper-parameters, which are often many. In this case, we argue that `GLIS` [10], `GLIS-r` [108], `GLISp` [11], `GLISp-r` [109], `C-GLIS`, `C-GLIS-r`, `C-GLISp` [156] and `C-GLISp-r` are relatively easy to compare (as opposed to, e.g., a Bayesian optimization procedure) since they rely on the same surrogate model for the cost function $f(\boldsymbol{x})$ of the GOP (2.1). In our benchmarks, we have chosen the same hyper-parameters for all the methods, whenever possible. This applies, for example, to the shape parameter ϵ_f and the radial function $\varphi_f(\cdot)$ of the surrogate model $\hat{f}_N(\boldsymbol{x})$ in (4.1). Vice-versa, for those algorithm-specific hyper-parameters (such as the trade-off weights of the several infill sampling criteria), we have used the settings suggested in the respective articles for `GLIS` [10], `GLISp` [11] and `C-GLISp` [156]. Concerning the cycling set Δ_{cycle} in (5.15) for `GLIS-r` [108], `GLISp-r` [109], `C-GLIS-r` and `C-GLISp-r`, we have mostly used $\Delta_{cycle} = \langle 0.95, 0.7, 0.35, 0\rangle$ since we have empirically found out that it results in a good compromise between exploration and exploitation (see Example 5.2). Δ_{cycle} also includes a zero term in compliance with the convergence result in Theorem 5.2. For unconstrained BBO and PBO, we also show the results obtained by setting $\Delta_{cycle} = \langle 0.95\rangle$ ("pure" exploitation) and $\Delta_{cycle} = \langle 0\rangle$ (pure exploration). Concerning the hyper-parameters of the PSVM classifier (in Section 6.1.3), we have used a Gaussian $\varphi_\Xi(\cdot)$ and a unitary shape parameter ϵ_Ξ for the model $m_{\Xi_N}(\boldsymbol{x})$ in (6.2), which are common choices (at least for machine learning problems [15, 153]). The threshold on the probability of Ξ-feasibility is set to $\delta = 0.5$ as in [9]. The last hyper-parameter, i.e. C_{SVM} for Problem (6.8), has been set to a relatively high value (that is $C_{SVM} = 10^6$) since no noise is present (see Assumption 2.4), making a classifier that behaves quite similarly to a hard margin SVM satisfactory. In BBO, no recalibration is performed for the surrogate model $\hat{f}_N(\boldsymbol{x})$ in (4.1) since neither `GLIS` [10] nor `C-GLIS` do so in their original formulation. Instead, in PBO, `GLISp` [11], `GLISp-r` [109], `C-GLISp` [156] and `C-GLISp-r` perform the recalibration at the same

[1]The software package is provided at `http://cse.lab.imtlucca.it/~bemporad/glis/`.

[2]Available at `http://www.norg.uminho.pt/aivaz/pswarm/`.

iterations and consider the same models $\mathcal{M}_f$ (see Section 4.4.2). In particular, as proposed in [11], we only tune the shape parameter ϵ_f and not the radial function $\varphi_f(\cdot)$ for $\hat{f}_N(\boldsymbol{x})$ in (4.1). Lastly, for what concerns the PSVM classifier used by `C-GLIS-r` and `C-GLISp-r`, no recalibration is performed since, at least empirically, the chosen hyper-parameters already behave quite well.
All the settings for the considered optimization procedures are reported in Appendix C.

7.1.2 Starting samples, budget and sample evaluations

For what concerns the number of samples generated by the initial experimental design (which is a LHD, see Section 2.4), i.e. N_{init}, we have selected it as follows:

- $N_{init} = 2 \cdot n$ for unconstrained BBO;
- $N_{init} = 4 \cdot n$ for unconstrained PBO;
- $N_{init} = 6 \cdot n$ for constrained BBO and PBO.

Note that N_{init} depends on the dimensionality n of the decision vector due to the fact that the exploration of the feasible region of the GOP (2.1) becomes progressively harder as n increases. Furthermore, for the unconstrained benchmarks, we start the preference-based optimization procedures from more samples than those used to initialize their black-box counterparts. That is because the preferences contained in $\mathcal{B}$ (3.9) and $\mathcal{S}$ (3.10) carry less information than the measures of $f(\boldsymbol{x})$ in $\mathcal{Y}$ (2.10); hence, we require more of the former to build an initial surrogate model[3]. Moreover, the number of initial samples for the constrained procedures is greater than N_{init} used by the unconstrained methods so that it is more likely to find a Ξ-feasible sample through the experimental design.
We point out that, in order to make the comparisons fair, *all the procedures are started from the same sets of samples*. Moreover, each benchmark optimization problem is solved $N_{trial} = 100$ times, starting from different sets of points, to collect statistically significant results. Lastly, the budget for all the algorithms is set to $N_{max} = 200$.
Overall, a total of 14400 trials have been performed, divided between the different benchmark optimization problems, the unconstrained/constrained BBO/PBO frameworks and the considered algorithms.
The next Remark clarifies how sample evaluations are carried out in the unconstrained and constrained PBO frameworks.

[3]For example, consider $n = 1$ and suppose $N_{init} = 2 \cdot n = 2$ for the PBO procedures. Generate $\mathcal{B}$ in (3.9) and $\mathcal{S}$ in (3.10) from Algorithm 7. Then, at the first iteration of the PBO methods, only $M = N_{init} - 1 = 1$ preference is available to build the surrogate model $\hat{f}_N(\boldsymbol{x})$ in (4.1).

Remark 7.1 (Sample evaluations in the preference-based setting)**.** *In this Chapter, which is purely devoted to comparing the proposed methods against the original ones, we do not consider a "real" decision-maker. Instead, the preferences are expressed directly from the cost functions* $f(\boldsymbol{x})$ *of the GOPs* (2.1) *and the* Ξ*-feasibility functions* $u_\Xi(\boldsymbol{x})$ *in* (2.4)*. Consider two samples* $\boldsymbol{x}_i, \boldsymbol{x}_j \in \Omega$*. In unconstrained PBO, we express a preference using the preference function* $\pi_{\gtrsim}(\boldsymbol{x}_i, \boldsymbol{x}_j)$ *in* (3.8)*, which we report here:*

$$\pi_{\gtrsim}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \begin{cases} -1 & \text{if } f(\boldsymbol{x}_i) < f(\boldsymbol{x}_j) \\ 0 & \text{if } f(\boldsymbol{x}_i) = f(\boldsymbol{x}_j) \\ 1 & \text{if } f(\boldsymbol{x}_i) > f(\boldsymbol{x}_j) \end{cases}. \tag{7.1}$$

Instead, in the constrained PBO setting, we express preferences by also penalizing the Ξ*-infeasibility [156]:*

$$\pi_{\gtrsim}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \begin{cases} -1 & \text{if either } u_\Xi(\boldsymbol{x}_i) = u_\Xi(\boldsymbol{x}_j) \text{ and } f(\boldsymbol{x}_i) < f(\boldsymbol{x}_j) \\ & \text{or } u_\Xi(\boldsymbol{x}_i) = 1, u_\Xi(\boldsymbol{x}_j) = 0 \\ 0 & \text{if } u_\Xi(\boldsymbol{x}_i) = u_\Xi(\boldsymbol{x}_j) \text{ and } f(\boldsymbol{x}_i) = f(\boldsymbol{x}_j) \\ 1 & \text{if either } u_\Xi(\boldsymbol{x}_i) = u_\Xi(\boldsymbol{x}_j) \text{ and } f(\boldsymbol{x}_i) > f(\boldsymbol{x}_j) \\ & \text{or } u_\Xi(\boldsymbol{x}_i) = 0, u_\Xi(\boldsymbol{x}_j) = 1 \end{cases}. \tag{7.2}$$

By proceeding as in (7.1) *and* (7.2)*, the preferences are always expressed consistently, allowing for a fair comparison of the PBO methods.*

Clearly, preference-based optimization procedures are designed to handle "real" decision-makers, which might not always be consistent in their choices. For this reason, in Chapter 8, we will show the performances of `GLISp-r` [109] and `C-GLISp-r` on a control systems case study and for which the author of this book plays the role of the DM.

7.1.3 Comparison criteria

We compare the performances of the optimization procedures on each benchmark optimization problem by means of convergence plots and data profiles [5]. Convergence plots depict the median, best and worst case performances over the N_{trial} trials and with respect to the cost function values achieved by $\boldsymbol{x_{best}}(N)$, as N increases. Data profiles show, for $1 \le N \le N_{max}$, how many among the N_{trial} trials of a benchmark optimization problem have been solved to a prescribed relative accuracy[4], defined as

[4]We refer to the indicator in (7.3) as relative accuracy instead of simply accuracy to avoid confusion with Definition 1.3.

[5]:

$$acc\left(N\right)=\frac{f\left(\boldsymbol{x_{best}}\left(N\right)\right)-f\left(\boldsymbol{x}_1\right)}{f^*-f\left(\boldsymbol{x}_1\right)}\cdot 100. \tag{7.3}$$

We point out that, typically, data profiles are used to visualize the performances of several algorithms on multiple benchmark optimization problems simultaneously. However, due to the stochastic nature of LHDs [91], here we will depict the data profiles for each method on each benchmark optimization problem, highlighting how the algorithms behave when started from different samples.

A benchmark optimization problem is said to be solved to a prescribed accuracy $t \in [0\%, 100\%]$ by some algorithm when $acc\left(N\right) > t$. Clearly, if $\boldsymbol{x_{best}}\left(N\right) = \boldsymbol{x}_i^*, \boldsymbol{x}_i^* \in \mathcal{X}^*$ in (2.3), then $acc\left(N\right) = 100\%$. We denote the number of samples required to reach a relative accuracy t as:

$$N_{acc>t} = \min_{1\leq N\leq N_{max}} N \qquad \text{such that } acc\left(N\right) > t. \tag{7.4}$$

Note that *it only makes sense to evaluate the relative accuracy* $acc\left(N\right)$ *in* (7.3) *when* $\boldsymbol{x_{best}}\left(N\right)$ *is actually* Ξ*-feasible*. That is because we could have $f\left(\boldsymbol{x_{best}}\left(N\right)\right) < f^*$ for some $\boldsymbol{x_{best}}\left(N\right) \notin \Xi$. Therefore, when black-box constraints are present, we have re-defined the relative accuracy in (7.3) to[5]:

$$acc\left(N\right)=\frac{f\left(\boldsymbol{x_{best}}\left(N\right)\right)-f\left(\boldsymbol{x}_{\tilde{i}}\right)}{f^*-f\left(\boldsymbol{x}_{\tilde{i}}\right)}\cdot 100, \qquad N \geq \tilde{i}, \tag{7.5}$$

$$\tilde{i} = \min_{i\in\mathbb{N}} i \qquad \text{such that } \boldsymbol{x}_i \in \Xi.$$

Differently from (7.3), the relative accuracy in (7.5) cannot exceed the value 100% (when $f\left(\boldsymbol{x_{best}}\left(N\right)\right) < f^*$ for some $\boldsymbol{x_{best}}\left(N\right) \notin \Xi$) and is better suited for the case $\Xi \neq \mathbb{R}^n$. Then, $N_{acc>t}$ in (7.4) can be re-written by using (7.5) instead of (7.3), obtaining:

$$N_{acc>t} = \min_{\tilde{i}\leq N\leq N_{max}} N \qquad \text{such that } acc\left(N\right) > t. \tag{7.6}$$

Overall, convergence plots and data profiles allow us to discern which algorithms are the most *efficient* and which are the most *robust* on the selected benchmark optimization problems. An algorithm `A` is efficient if it exhibits fast convergence speeds, i.e. it is able to find a candidate sample $\boldsymbol{x_{best}}\left(N\right)$ for which $\left|f\left(\boldsymbol{x_{best}}\left(N\right)\right) - f^*\right|$ is sufficiently small and N is low (cf. Definition 1.4). If `A` is efficient on some benchmark optimization problem then, in its corresponding convergence plot, the curve associated to the median values of $f\left(\boldsymbol{x_{best}}\left(N\right)\right)$ approaches the global minimum f^* quite quickly. Similarly, its respective data profile is a curve that raises rapidly. When it comes to robustness, we analyze how many among the N_{trial} trials for a given benchmark optimization problem are solved to

[5]Note that, due to how surrogate-based methods work, $\boldsymbol{x}_i \in \Omega, \forall i : 1 \leq i \leq N_{max}$. Hence, we do not need to specify explicitly that $\boldsymbol{x}_{\tilde{i}} \in \Xi \cap \Omega$ in (7.5). Similarly, for unconstrained BBO and PBO, we always have $\boldsymbol{x}_1 \in \Xi \cap \Omega$ in (7.3).

a prescribed relative accuracy $t \in [0\%, 100\%]$. This can be easily seen in the data profiles of A: if, for $N \to N_{max}$, the algorithm is able to solve most of the considered N_{trial} trials, then A is robust. Usually, when A fails to reach $acc\,(N) > t$ on at least one among the N_{trial} trials, the worst case values of $f\,(\boldsymbol{x}_{\boldsymbol{best}}\,(N))$ shown in the corresponding convergence plots of A are much higher than the median values of the cost function achieved by the best candidates. Typically, no optimization algorithm is both efficient and robust on all optimization problems. Rather, a trade-off between these two properties must be made when designing A [142]. For example, the Grid Search algorithm (Section 1.2.2) is quite robust (since it is bound to explore the whole feasible set Ω) but extremely inefficient. At the same time, assuming that the gradients of the cost function and (possibly) the constraints functions of the GOP (2.1) are available, a derivative-based procedure, such as the Newton algorithm, is more likely to be efficient but less robust (in a sense that it converges rapidly to a solution that might just be a local minimizer of the GOP (2.1)). That is because the solutions found by derivative-based procedures depend highly on their starting points (see the definition of basin of attraction, i.e. Definition 1.5).

We also consider an additional indicator, $d_{rel}\,(N_{max})$, which corresponds to the *relative distance between the global minimizer(s) of the GOP* (2.1) *and the solution found by an optimization algorithm A*:

$$d_{rel}\,(N_{max}) = \frac{\min_{\boldsymbol{x}_i^* \in \mathcal{X}^*} \left\| \boldsymbol{x}_{\boldsymbol{best}}\,(N_{max}) - \boldsymbol{x}_i^* \right\|_2}{\left\| \boldsymbol{u} - \boldsymbol{l} \right\|_2} \cdot 100. \tag{7.7}$$

Note that, differently from $acc\,(N)$ in (7.3), $d_{rel}\,(\cdot)$ in (7.7) is not monotonic and therefore it only makes sense to evaluate it when the budget is exhausted.

Lastly, the black-box and preference-based optimization procedures are compared based on the *average computational times* required to generate and evaluate N_{max} samples. We remark that, in this case, the computational overheads linked to the sample evaluations are negligible. That is due to the fact that the considered benchmark global optimization problems only include analytical functions, i.e. there is no need to run any expensive simulation to measure $f\,(\cdot)$ or $u_{\Xi}\,(\cdot)$. Therefore, the reported average computational times are practically the execution times of the optimization procedures.

7.2 Unconstrained black-box and preference-based optimization

Figure 34 and Figure 35 depict the convergence plots and the data profiles ($acc\,(N) > 95\%$) of GLIS [10], C-GLIS and GLIS-r [108] on the unconstrained black-box optimization problems. Instead, Figure 36 and Figure 37 show the convergence plots and the data profiles ($acc\,(N) > 95\%$) of GLISp [11], C-GLISp [156] and GLISp-r [109] on the unconstrained preference-based optimization problems. The results achieved by GLIS-r [108] and GLISp-r [109] for the "pure" exploitation

($\Delta_{cycle} = \langle 0.95 \rangle$) and the pure exploration ($\Delta_{cycle} = \langle 0 \rangle$) hyper-parameter settings are reported only in the data profiles, to make the graphs more readable.

Table 2 reports the number of samples required to reach a median (over the N_{trial} trials) relative accuracy $acc(N)$ of 95% (as in (7.3) and (7.4)). Instead, Table 3 shows the performances of the algorithms with respect to the $d_{rel}(N_{max})$ indicator in (7.7) (median-wise). Lastly, Table 4 reports the average computational times required by the optimization procedures to generate and evaluate $N_{max} = 200$ samples.

From these benchmarks we gather that:

- *In the black-box framework,* `GLIS-r` *[108] ($\Delta_{cycle} = \langle 0.95, 0.7, 0.35, 0 \rangle$) achieves similar performances to* `GLIS` *[10] on several benchmark optimization problems*, without relying on the IDW variance function $s_N(\boldsymbol{x})$ in (4.19). That is the case for the `bemporad` [10], `levi 13` [92], `adjiman` [62], `rosenbrock` [62] and `step 2` [62] benchmarks. Instead, `GLIS-r` [108] is the clear winner (both in terms of efficiency and robustness) for the `gramacy and lee` [53] problem, while `GLIS` [10] outperforms all the other procedures on the `bukin 6` [62] benchmark. Lastly, the `ackley` [62] and the `salomon` [62] benchmarks prove to be quite hard for all the considered algorithms. For what concerns `C-GLIS`, in our trials it is often slower than `GLIS` [10] and `GLIS-r` [108] ($\Delta_{cycle} = \langle 0.95, 0.7, 0.35, 0 \rangle$) since it prioritizes exploration quite a lot during its early iterations (due to how $z_N(\boldsymbol{x})$ in (4.18) is defined).

- *In the preference-based framework,* `GLISp-r` *[109] ($\Delta_{cycle} = \langle 0.95, 0.7, 0.35, 0 \rangle$) can be notably more robust than* `GLISp` *[11] without excessively compromising its convergence speed.* On several occasions, the latter algorithm gets stuck on local minima of the benchmark GOPs (2.1). That is particularly evident on the `bemporad` [10] and `gramacy and lee` [53] benchmarks, in which cases `GLISp` [11] solves, respectively, only 70% and 31% of the N_{trial} trials. Instead, `GLISp-r` [109] ($\Delta_{cycle} = \langle 0.95, 0.7, 0.35, 0 \rangle$) is able to solve all of them to the prescribed relative accuracy. As further proof, the best candidates found by all the considered procedures when solving the `gramacy and lee` [53] benchmark are depicted in Figure 38. In this case, $f(\boldsymbol{x})$ is highly multimodal and `GLISp` [11] often fails to find the global minimizer $\boldsymbol{x}^*$. That is due to the shortcomings of $z_N(\boldsymbol{x})$ in (4.17) (see Section 5.1), which compromise the exploratory capabilities of the method. As a matter of fact, on the `gramacy and lee` [53] benchmark, even `GLISp-r` [109] with $\Delta_{cycle} = \langle 0 \rangle$ (pure exploration) converges to $\boldsymbol{x}^*$ relatively quickly. Conversely, `GLISp` [11] shines when exploitation is better suited for the benchmark GOP (2.1) at hand. That is particularly relevant for the `bukin 6` [62] problem, on which both `GLISp` [11] and

GLISp-r [109] with $\Delta_{cycle} = \langle 0.95 \rangle$ ("pure" exploitation) perform quite well. When it comes to unconstrained PBO, C-GLISp [156] is as robust as GLISp-r [109] ($\Delta_{cycle} = \langle 0.95, 0.7, 0.35, 0 \rangle$) but it is the least efficient among the analyzed procedures. Lastly, the ackley [62] and the salomon [62] benchmarks, which were already quite hard for the BBO procedures, prove to be even more challenging in the PBO framework.

- Both in BBO and PBO, the pure exploration strategy (i.e. GLIS-r [108] and GLISp-r [109] with $\Delta_{cycle} = \langle 0 \rangle$) performs poorly, even for $n = 2$. When $n = 5$, it is unable solve any problem (in fact, the data profiles stay flat after the initial experimental design). Only for $n = 1$ the pure exploration strategy is quite robust and relatively efficient.

- Vice-versa, a pure exploitatory approach (i.e. GLIS-r [108] and GLISp-r [109] with $\Delta_{cycle} = \langle 0.95 \rangle$), although not necessarily globally convergent (see Theorem 5.2), can actually be successful on some benchmark GOPs (2.1), both in BBO and PBO. Often, such strategy exhibits a slightly lower $N_{acc>95\%}$ (median-wise) than the other procedures, see Table 2. *Notably, the data profiles of GLISp-r [109] ($\Delta_{cycle} = \langle 0.95 \rangle$) can be quite similar to those of GLISp [11]. Therefore, we could say that GLISp [11] often amounts to pure exploitation.* Even the best candidate samples returned by GLISp-r [109] ($\Delta_{cycle} = \langle 0.95 \rangle$) and GLISp [11] when solving the gramacy and lee [53] benchmark are alike, see Figure 38.

- Overall, if we take a look only at GLIS-r [108] and GLISp-r [109] with $\Delta_{cycle} = \langle 0.95, 0.7, 0.35, 0 \rangle$, we can see that, *if the preferences are expressed consistently (as in Remark 7.1), PBO methods can achieve similar results to those of their BBO counterparts despite using less information on the cost function $f(\boldsymbol{x})$ of the GOP* (2.1). This is highlighted both by $N_{acc>95\%}$ in Table 2 and $d_{rel}(N_{max})$ in Table 3.

- *The main disadvantage of GLIS-r [108] and GLISp-r [109] compared to the original procedures is the increased computational time*, as reported in Table 4. That is due to the computational overhead of Algorithm 13, which generates the augmented sample set $\mathcal{X}_{aug}$ for the proposed procedures by performing K-means clustering[6]. The difference in computational times is quite pronounced for the BBO procedures; instead, for PBO, the run times of GLISp [11], C-GLISp [156] and GLISp-r [109] become more similar, mostly due to the LOOCV performed to recalibrate the shape parameter ϵ_f of the surrogate model $\hat{f}_N(\boldsymbol{x})$ in (4.1) (see Section 4.4.2).

[6]Also, we have written the code for GLIS-r [108] and GLISp-r [109] from scratch, obtaining a software package that is quite different from that of GLIS [10], GLISp [11], C-GLIS and C-GLISp [156], and (possibly) less computationally efficient.

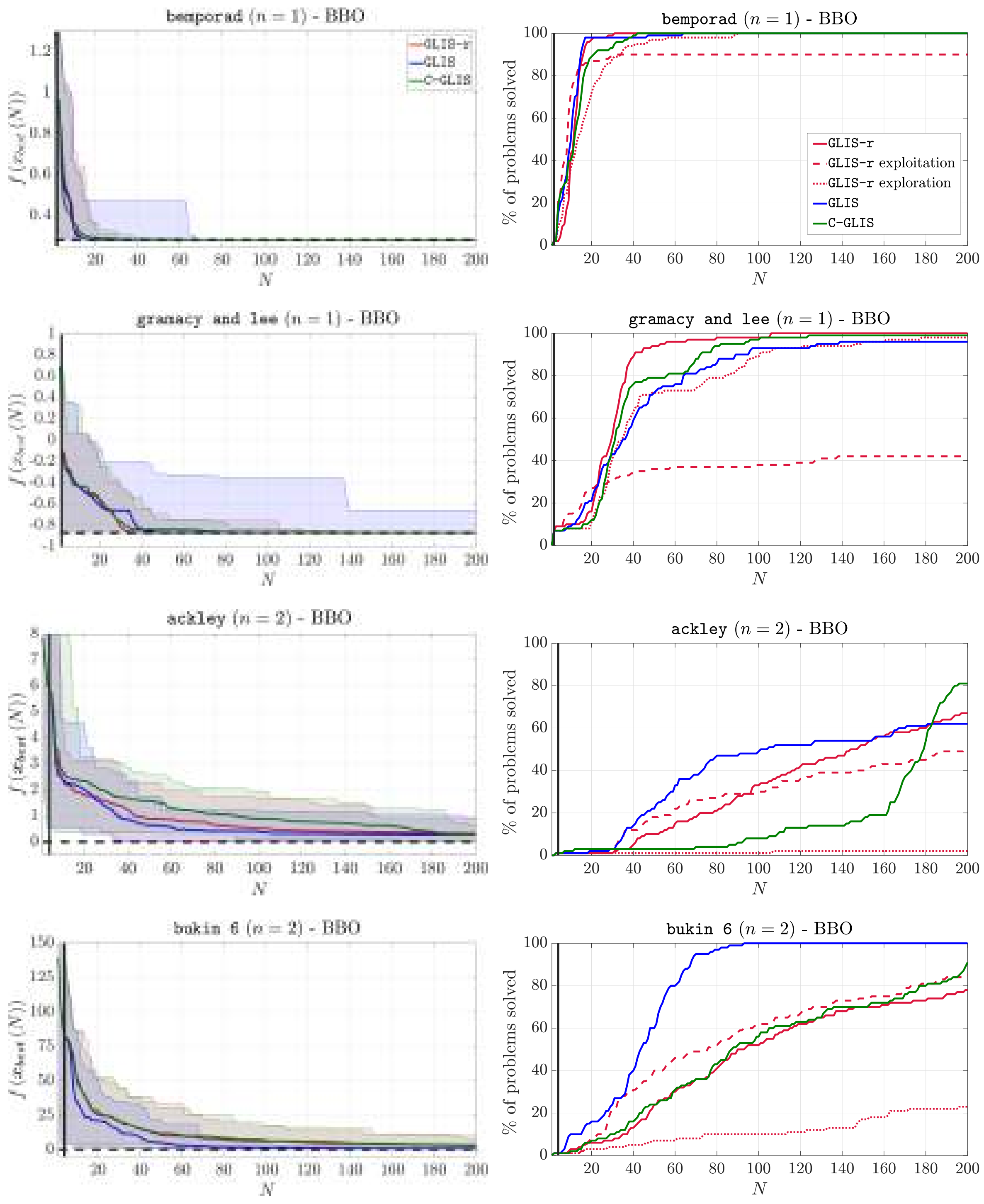


Figure 34: Performances achieved by the different unconstrained black-box optimization algorithms on the benchmark GOPs (2.1): convergence plots on the left and data profiles ($acc(N) > 95\%$) on the right. `GLIS-r` [108] (with $\Delta_{cycle} = \langle 0.95, 0.7, 0.35, 0 \rangle$) is depicted in red, `GLIS` [10] in blue and `C-GLIS` in green. The dashed black-line in the convergence plots represents f^*. We also show the number of initial samples, N_{init}, with a black vertical line. The results obtained by `GLIS-r` [108] with $\Delta_{cycle} = \langle 0.95 \rangle$ (dashed red line) and `GLIS-r` [108] with $\Delta_{cycle} = \langle 0 \rangle$ (dotted red line) are shown only in the data profiles.

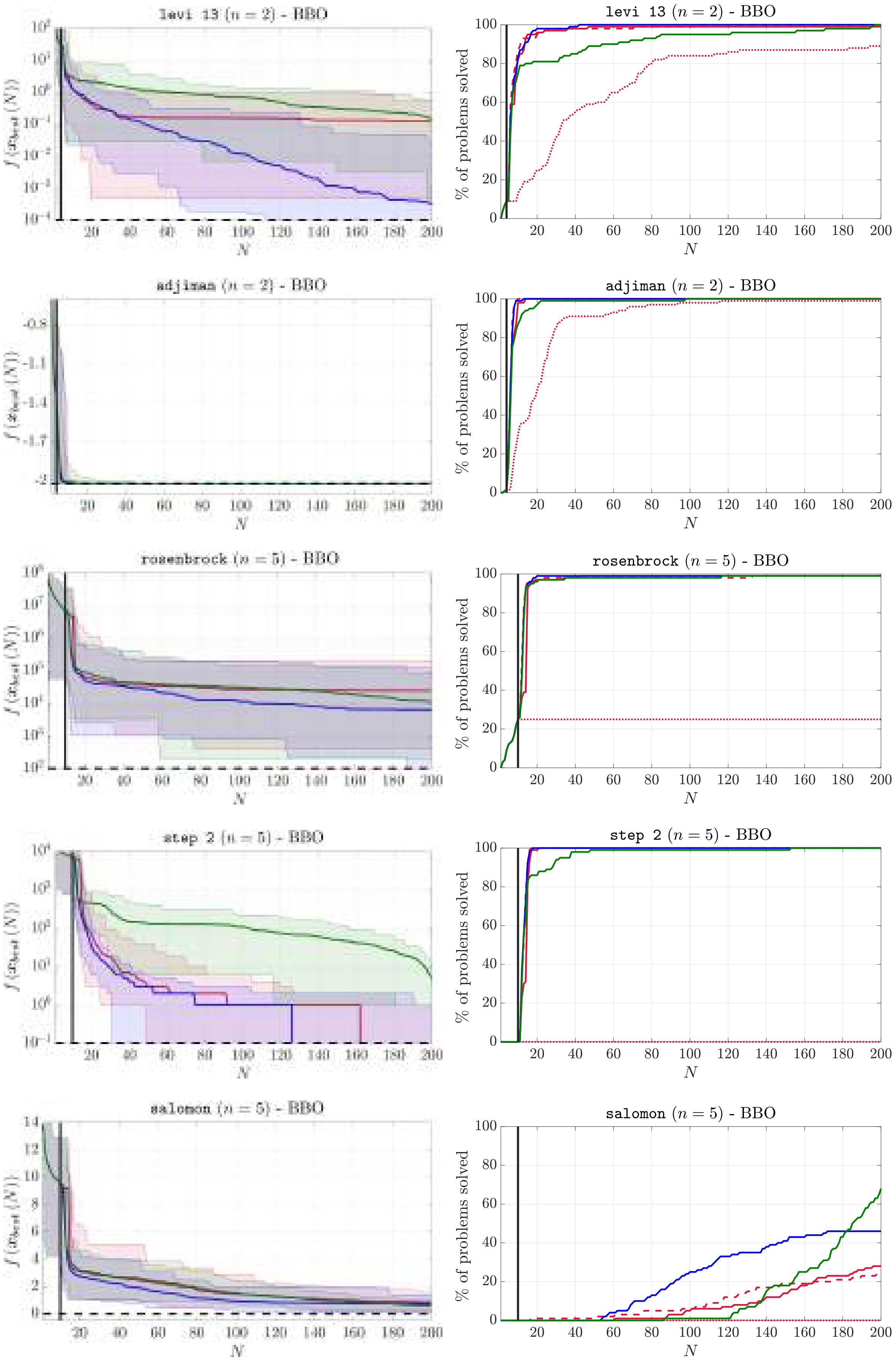


Figure 35: Figure 34 cont'd (unconstrained BBO).

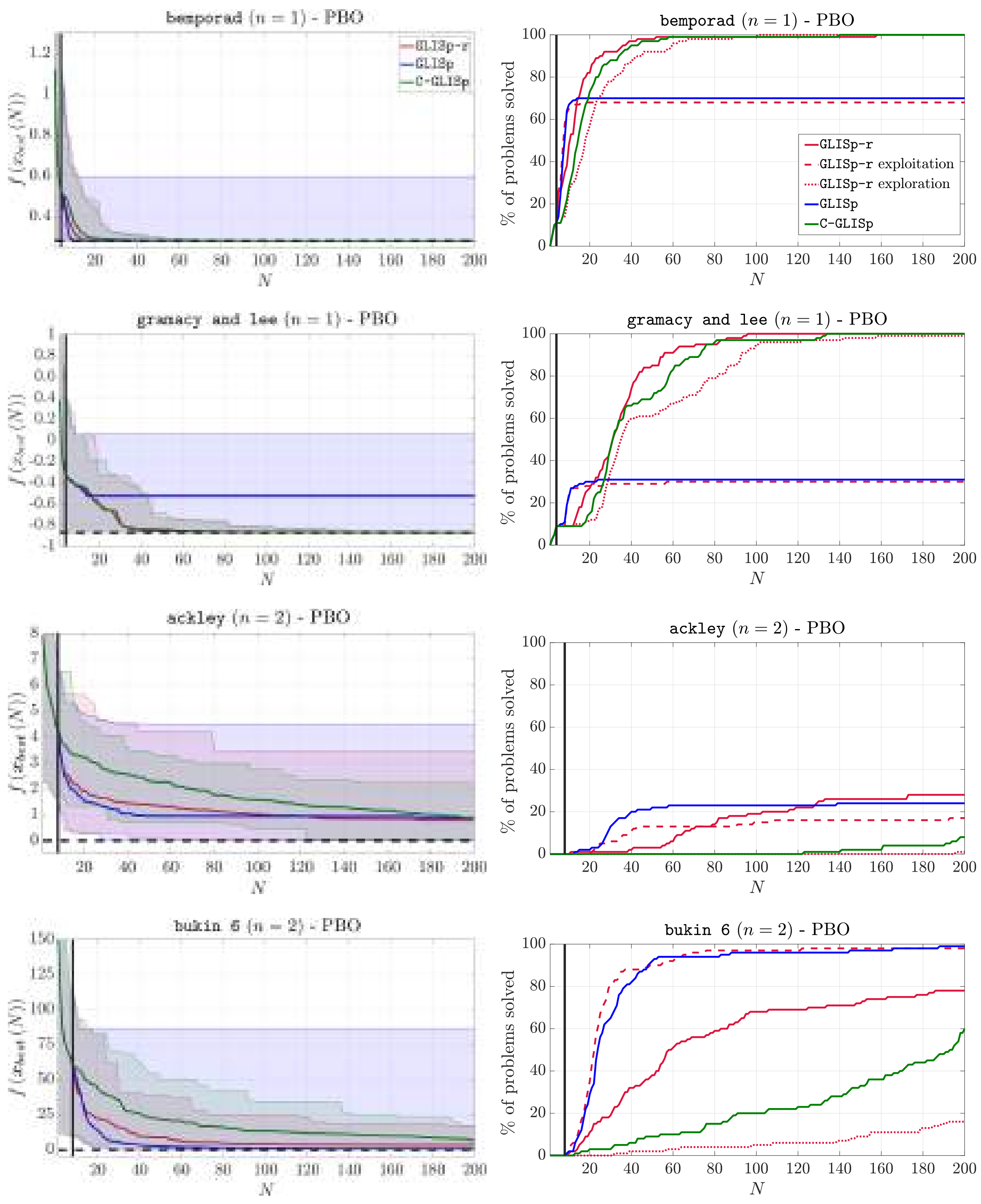


Figure 36: Performances achieved by the different unconstrained preference-based optimization algorithms on the benchmark GOPs (2.1)**: convergence plots on the left and data profiles** $(acc\,(N) > 95\%)$ **on the right. `GLISp-r` [109] (with $\Delta_{cycle} = \langle 0.95, 0.7, 0.35, 0 \rangle$) is depicted in red, `GLISp` [11] in blue and `C-GLISp` [156] in green. The dashed black-line in the convergence plots represents f^*. We also show the number of initial samples, N_{init}, with a black vertical line. The results obtained by `GLISp-r` [109] with $\Delta_{cycle} = \langle 0.95 \rangle$ (dashed red line) and `GLISp-r` [109] with $\Delta_{cycle} = \langle 0 \rangle$ (dotted red line) are shown only in the data profiles.**

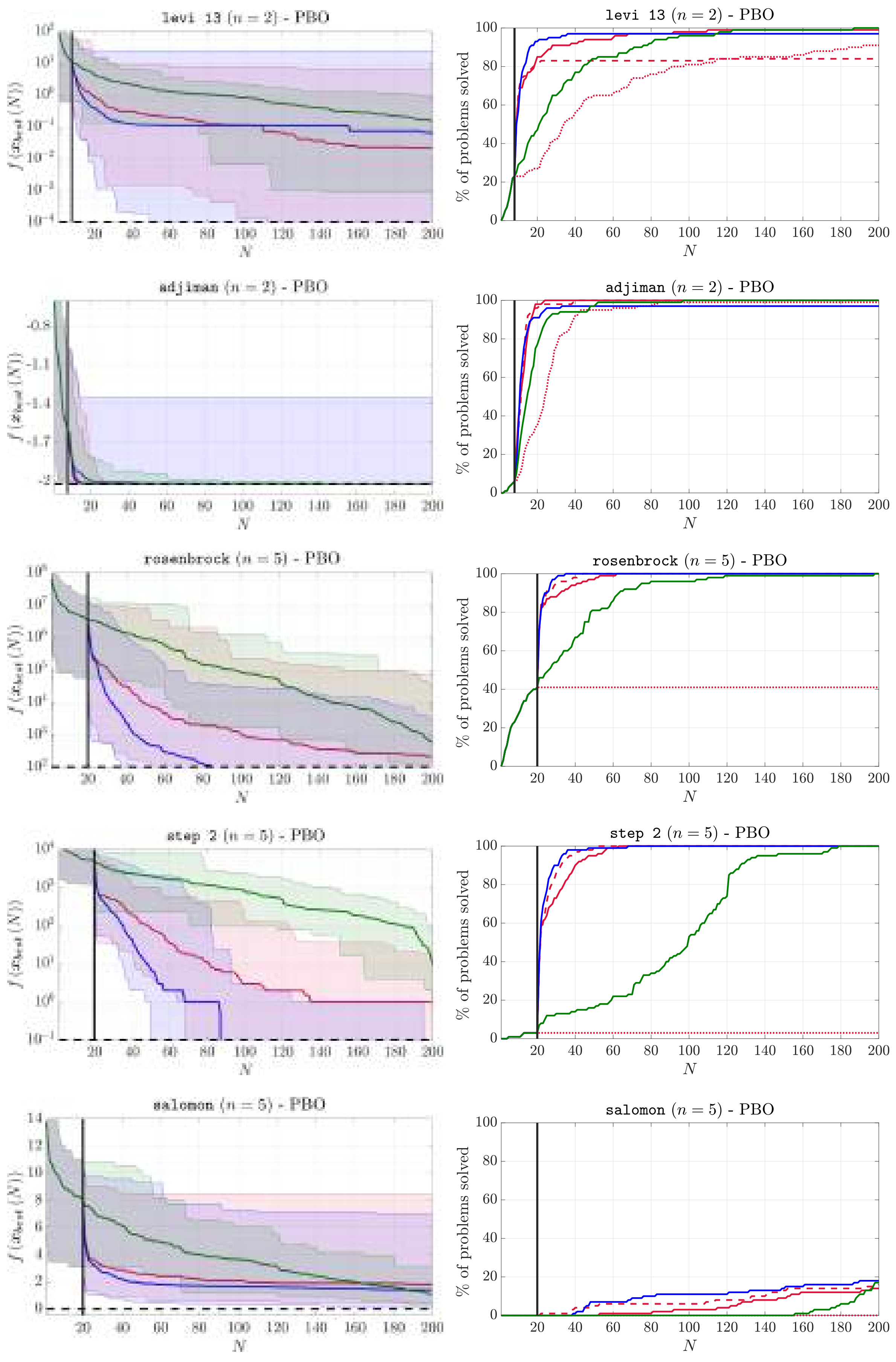


Figure 37: Figure 36 cont’d (unconstrained PBO).

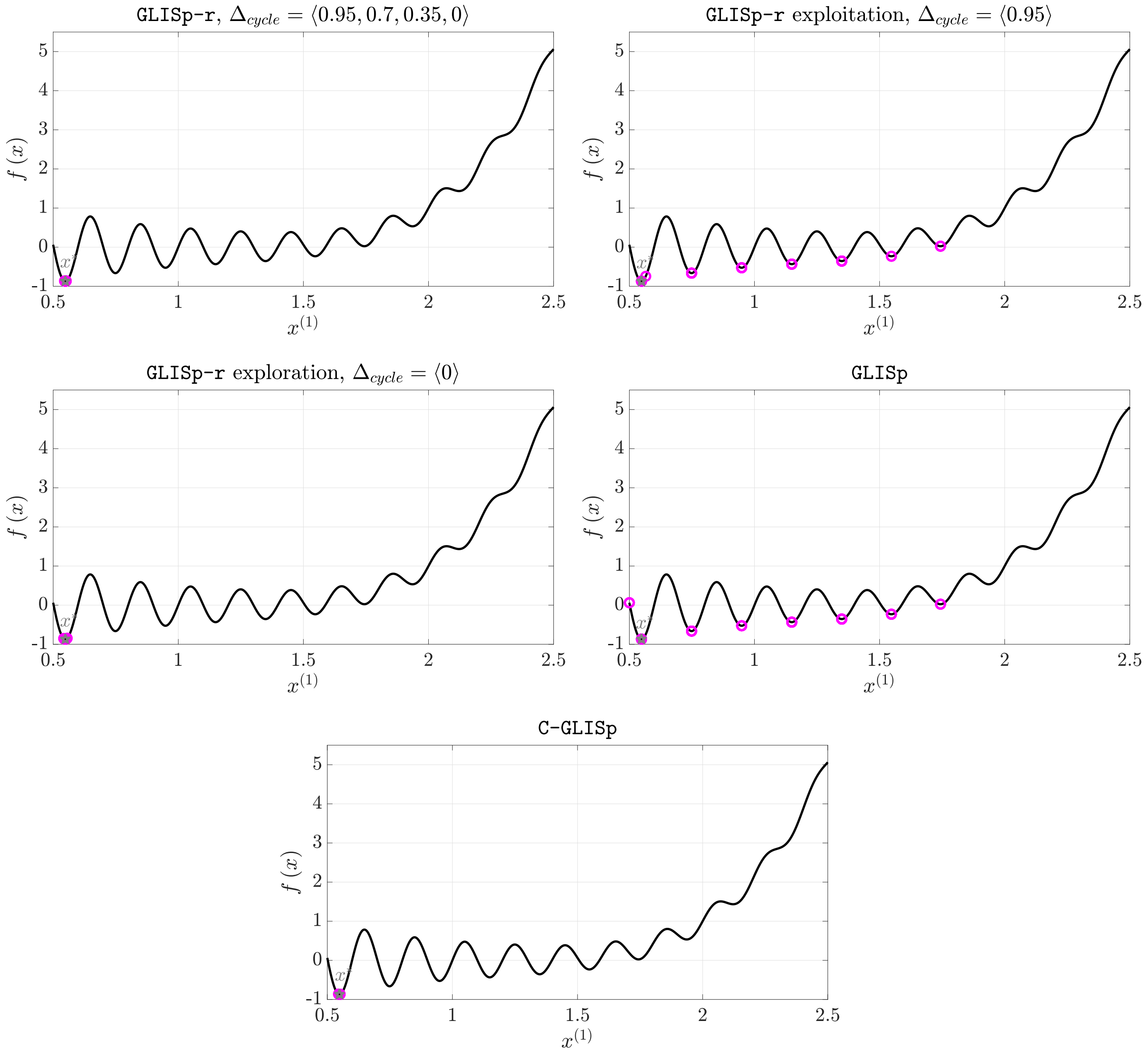


Figure 38: Best samples found by the unconstrained preference-based optimization procedures on the `gramacy and lee` [53] benchmark. Each point in magenta represents $x_{best}\ (N_{max})$ for the corresponding trial. The grey star depicts the global minimizer of the GOP (2.1).

Table 2: Number of samples required by the unconstrained black-box and preference-based optimization algorithms to solve the benchmark GOPs (2.1) **to a relative accuracy of** 95% **(median-wise). The acronym `n.r.` stands for "not reached". The best results are highlighted in bold font.**

			$N_{acc>95\%}$				
Benchmark problem	n	**Optimization**	`GLIS\p`	`C-GLIS\p`	`GLIS\p-r`	`GLIS\p-r` exploitation	`GLIS\p-r` exploration
`bemporad` [10]	1	black-box	11	12	12	**9**	14
		preference-based	**8**	15	11	**8**	18
`gramacy and lee` [53]	1	black-box	37	32	**30**	`n.r.`	35
		preference-based	`n.r.`	32	**31**	`n.r.`	36
`ackley` [62]	2	black-box	**101**	180	144	`n.r.`	`n.r.`
		preference-based	`n.r.`	`n.r.`	`n.r.`	`n.r.`	`n.r.`
`bukin 6` [62]	2	black-box	**44**	88	90	76	`n.r.`
		preference-based	24	193	58	**23**	`n.r.`
`levi 13` [92]	2	black-box	**6**	7	**6**	**6**	34
		preference-based	**9**	22	**9**	**9**	34
`adjiman` [62]	2	black-box	**6**	7	7	**6**	20
		preference-based	**11**	15	12	12	24
`rosenbrock` [62]	5	black-box	13	13	15	**12**	`n.r.`
		preference-based	**21**	26	**21**	**21**	`n.r.`
`step 2` [62]	5	black-box	**13**	**13**	15	**13**	`n.r.`
		preference-based	**22**	100	**22**	**22**	`n.r.`
`salomon` [62]	5	black-box	`n.r.`	**186**	`n.r.`	`n.r.`	`n.r.`
		preference-based	`n.r.`	`n.r.`	`n.r.`	`n.r.`	`n.r.`

Table 3: Median relative distances from the global minimizer(s) of the considered benchmark GOPs (2.1) **achieved by the unconstrained black-box and preference-based optimization algorithms. The best results are highlighted in bold font.**

			$d_{rel}\ (N_{max})$				
Benchmark problem	n	**Optimization**	`GLIS\p`	`C-GLIS\p`	`GLIS\p-r`	`GLIS\p-r` exploitation	`GLIS\p-r` exploration
`bemporad` [10]	1	black-box	**0.00%**	0.01%	0.04%	**0.00%**	0.13%
		preference-based	**0.00%**	0.04%	0.04%	**0.00%**	0.12%
`gramacy and lee` [53]	1	black-box	**0.00%**	0.02%	0.02%	10.00%	0.11%
		preference-based	20.02%	0.03%	**0.01%**	20.02%	0.12%
`ackley` [62]	2	black-box	1.01%	0.98%	**0.94%**	1.39%	3.13%
		preference-based	3.19%	**2.09%**	2.24%	4.10%	3.30%
`bukin 6` [62]	2	black-box	15.57%	17.42%	**14.27%**	17.93%	24.52%
		preference-based	17.46%	21.13%	19.04%	**16.00%**	19.14%
`levi 13` [92]	2	black-box	**0.03%**	0.69%	0.63%	0.27%	2.19%
		preference-based	0.71%	0.66%	**0.20%**	1.16%	2.28%
`adjiman` [62]	2	black-box	**0.00%**	0.04%	**0.00%**	**0.00%**	1.03%
		preference-based	**0.00%**	0.08%	**0.00%**	**0.00%**	1.06%
`rosenbrock` [62]	5	black-box	7.05%	7.96%	6.99%	**5.93%**	20.12%
		preference-based	**1.73%**	2.49%	2.49%	1.75%	16.87%
`step 2` [62]	5	black-box	0.29%	0.55%	**0.28%**	0.29%	18.69%
		preference-based	**0.28%**	0.73%	0.31%	**0.28%**	15.89%
`salomon` [62]	5	black-box	1.56%	**1.18%**	1.73%	1.92%	19.45%
		preference-based	3.07%	**2.29%**	3.86%	3.87%	15.35%

Table 4: Average execution times of the unconstrained black-box and preference-based optimization algorithms. The best results are highlighted in bold font.

			Average execution times [sec]				
Benchmark problem	n	**Optimization**	`GLIS\p`	`C-GLIS\p`	`GLIS\p-r`	`GLIS\p-r` exploitation	`GLIS\p-r` exploration
`bemporad` [10]	1	black-box	**8.0**	7.7	40.4	40.1	42.9
		preference-based	79.8	**57.5**	79.4	109.7	71.4
`gramacy and lee` [53]	1	black-box	**5.4**	6.7	30.7	30.5	35.2
		preference-based	74.4	**58.1**	67.5	90.8	63.8
`ackley` [62]	2	black-box	**22.7**	22.8	67.2	53.5	93.0
		preference-based	92.1	**70.3**	127.9	122.4	121.1
`bukin 6` [62]	2	black-box	**25.0**	29.7	65.6	59.3	87.9
		preference-based	84.5	**73.3**	115.0	114.6	123.6
`levi 13` [92]	2	black-box	**22.7**	24.1	69.9	57.7	88.7
		preference-based	82.0	**69.3**	126.8	118.2	120.6
`adjiman` [62]	2	black-box	**22.6**	23.8	61.1	59.7	88.0
		preference-based	105.4	**71.6**	146.1	124.0	117.5
`rosenbrock` [62]	5	black-box	57.5	**55.5**	101.2	135.7	92.3
		preference-based	133.3	107.1	190.2	222.3	**100.2**
`step 2` [62]	5	black-box	**52.9**	58.1	106.6	129.8	85.9
		preference-based	148.1	104.2	199.6	216.7	**98.9**
`salomon` [62]	5	black-box	**49.8**	50.4	101.8	127.0	87.6
		preference-based	137.7	103.5	183.4	219.1	**100.8**

7.3 Constrained black-box and preference-based optimization

Figure 39 and Figure 40 depict the convergence plots and the data profiles ($acc(N) > 95\%$) of the constrained black-box optimization procedures, namely `C-GLIS` and `C-GLIS-r`. For the latter method, we compare the performances of the revisited PSVM classifier (Section 6.1.3) and those of the IDWI function in (4.16), when used to approximate the Ξ-feasibility function $u_{\Xi}(\boldsymbol{x})$ in (2.4). For the sake of clarity, we refer to these two formulations as `C-GLIS-r` PSVM and `C-GLIS-r` IDWI respectively. Instead, Figure 41 and Figure 42 show the convergence plots and the data profiles ($acc(N) > 95\%$) of the constrained preference-based optimization algorithms, namely `C-GLISp` [156] and the two formulations of `C-GLISp-r` (i.e. `C-GLISp-r` PSVM and `C-GLISp-r` IDWI). To make the convergence plots easier to read, we use a dashed line whenever an algorithm has yet to find a Ξ-feasible sample in all the N_{trial} trials. Then, as as soon as all the best candidates for the N_{trial} trials are Ξ-feasible, we switch to a continuous line. Note that, at each N such that $1 \leq N \leq N_{max}$, the median, best and worst case values of $f(\boldsymbol{x_{best}}(N))$ are computed only from those $\boldsymbol{x_{best}}(N) \in \Xi$, otherwise we could have $f(\boldsymbol{x_{best}}(N)) < f^*$ for some $\boldsymbol{x_{best}}(N) \notin \Xi$, making the convergence plots unreliable. However, we remark that as long as an algorithm has yet to find a Ξ-feasible candidate in all N_{trial} trials, the median, best and worst case values of $f(\boldsymbol{x_{best}}(N))$ are not necessarily monotone decreasing (see for example the `camel six humps constrained` benchmark GOP in Figure 40). For what concerns the data profiles, we point out that we deem a problem solved when $acc(N) > 95\%$ and $\boldsymbol{x_{best}}(N) \in \Xi$, consistently with (7.5).

Table 5 reports the number of samples required to reach a median (over the N_{trial} trials) relative accuracy $acc(N)$ of 95%. $acc(N)$ and $N_{acc>95\%}$ are defined, respectively, in (7.5) and (7.6). We have chosen to report $N_{acc>95\%} \geq \tilde{N}$, where $\tilde{N}$ is the number of samples required by an algorithm `A` to make $\boldsymbol{x_{best}}(N) \in \Xi, \forall N : \tilde{N} \leq N \leq N_{max}$ and for all N_{trial} trials of a benchmark optimization problem. Roughly speaking, we start evaluating the indicator $N_{acc>95\%}$ only when `A` finds a Ξ-feasible best candidate on all N_{trial} trials. The rationale behind this choice is that it makes the results reported in Table 5 more statistically significant. If that were not the case, $N_{acc>95\%}$ might be computed only on a fraction of the N_{trial} trials (only those for which $\boldsymbol{x_{best}}(N) \in \Xi$), preventing the indicator from capturing the overall behavior of `A` on the chosen benchmark optimization problem.

Table 6 shows the performances of the algorithms with respect to the $d_{rel}(N_{max})$ indicator in (7.7) (median-wise). Similarly to $N_{acc>95\%}$, for constrained BBO and PBO we only consider those $\boldsymbol{x_{best}}(N_{max}) \in \Xi$ when computing $d_{rel}(N_{max})$ and report the latter indicator if and only if `A` has found a Ξ-feasible sample on all N_{trial} trials. Lastly, Table 7 reports the average computational times required by the optimization procedures to generate and evaluate $N_{max} = 200$ samples.

We now discuss the results achieved by the procedures on the benchmark GOPs (2.1):

- Both in constrained BBO and PBO, *the proposed methods `C-GLIS-r` and `C-GLISp-r`, equipped with either the PSVM classifier or the IDW interpolant, are more robust and efficient than `C-GLIS` and `C-GLISp` [156] on the lower-dimensional benchmark GOPs* (2.1) ($n = 1$ and $n = 2$). Instead, on the `welded beam design` [78] ($n = 4$) and the `himmelblau` [78] ($n = 5$) benchmarks, `C-GLIS` (respectively, `C-GLISp` [156]) achieves performances that are similar to either `C-GLIS-r` (`C-GLISp-r`) PSVM or `C-GLIS-r` (`C-GLISp-r`) IDWI. The only exception is the black-box optimization of the `himmelblau` [78] problem, on which `C-GLIS-r` IDWI underperforms (both in terms of robustness and efficiency), while `C-GLIS-r` PSVM is the fastest, albeit being slightly less robust than `C-GLIS`. Instead, as we will see shortly, the `step 2 constrained` ($n = 5$) benchmark requires a more extensive discussion.

- For what concerns the `townsend` [1] benchmark, it seems like, apart from `C-GLIS-r` IDWI in the BBO framework, all the other methods are unable to find good solutions for the problem. As a matter of fact, the median values of the cost function in the convergence plots "flatline" while the curves reported in the data profiles stay below 10%. In reality, we have empirically found out that even if we were to employ the `PSWARM` [72] algorithm to solve the benchmark GOP (2.1) directly, it would not always be successful. Therefore, it is quite unlikely that any of the considered surrogate-based methods (which rely on the `PSWARM` [72] procedure) are able to solve the `townsend` [1] benchmark.

- The `camel six humps constrained` [156] benchmark proves to be quite hard for all the analyzed algorithms. That is because the size of the Ξ-feasible region of the GOP (2.1) is small compared to Ω. As a matter of fact, the algorithms take many iterations to find an initial Ξ-feasible sample. In Figure 40, we can see that `C-GLIS` takes (roughly) $N = 85$ sample evaluations to gain Ξ-feasibility on all N_{trial} trials. Instead, `C-GLIS-r` PSVM and `C-GLIS-r` IDWI, although more efficient, can require more than half of the budget to find an $\boldsymbol{x_{best}}(N) \in \Xi$. In the constrained PBO framework, `C-GLISp` [156] fails to find a Ξ-feasible sample on 2/100 trials, as depicted in Figure 43. That is not the case for `C-GLISp-r` PSVM and `C-GLISp-r` IDWI, which are always successful in doing so.

- The `sasena 2` [125] benchmark is also challenging but for a different reason: the problem has a local and a global minimizer located in separate, disconnected regions of Ξ (see Example 6.4). In many trials, depending on the locations of the initial samples, the algorithms get stuck in a neighborhood of the local minimizer, hence why the convergence plots "flatline". Notably, in BBO, `C-GLIS-r` IDWI is able to get close to the global minimizer of the `sasena 2` [125] benchmark on more than half of the N_{trial} trials.

- The behaviors of the several optimization procedures on the `step 2 constrained` benchmark are quite interesting. The proposed methods are unable to find a Ξ-feasible candidate on all trials (more precisely, only $63/100$ trials are successful), but when they do `C-GLIS-r` PSVM (respectively, `C-GLISp-r` PSVM) and `C-GLIS-r` IDWI (`C-GLISp-r` IDWI) are more efficient than `C-GLIS` (`C-GLISp` [156]), as highlighted by their respective convergence plots and data profiles. Vice-versa, `C-GLIS` and `C-GLISp` [156] return an $\boldsymbol{x_{best}}\left(N_{max}\right) \in \Xi$ on all N_{trial} trials, as depicted in Figure 44, but are much slower. Figure 44 also shows the level curves of the cost function for the `step 2 constrained` benchmark when $n = 2$. We motivate the behaviors of the algorithms as follows:

  - Regardless of the choice of the surrogate model for the probability of Ξ-feasibility, whenever no Ξ-feasible samples are present, `C-GLIS-r` and `C-GLISp-r` follow a pure exploration approach, proposing new candidate samples by solving Problem (6.34). In Example 6.3, we have already seen how, as n increases, minimizing $z_N\left(\boldsymbol{x}\right)$ in (4.17) can be quite inefficient in seeking Ξ-feasible candidates, hence why the proposed methods find an $\boldsymbol{x_{best}}\left(N\right) \in \Xi$ only on $63/100$ trials.

  - `C-GLIS` and `C-GLISp` [156] do not vary their infill sampling criteria based on how many Ξ-feasible/Ξ-infeasible samples are present in $\mathcal{X}$ (2.9). However, in practice, when $\boldsymbol{x}_i \notin \Xi, \forall \boldsymbol{x}_i \in \mathcal{X}$, resulting in $u_i = 0, \forall u_i \in \mathcal{U}_\Xi$, the IDW interpolant in (4.16) does not penalize any region of Ω. That is because, due to Property 2 in Proposition 4.3, we have:

$$p_N\left(\boldsymbol{x} \in \Xi\right) = 0, \quad \forall \boldsymbol{x} \in \Omega.$$

    Therefore, the acquisition functions in (4.20c) and (4.20d) respectively amount to $a_N\left(\boldsymbol{x}\right)$ in (4.20a) and (4.20b), which are those of `GLIS` [10] and `GLISp` [11] (unconstrained BBO and PBO), but with $z_N\left(\boldsymbol{x}\right)$ in (4.18) instead of (4.17). When dealing with the `step 2 constrained` benchmark, this approach proves to be successful since the unconstrained minimization of $f\left(\boldsymbol{x}\right)$ can lead `C-GLIS` and `C-GLISp` [156] to sample inside the Ξ-feasible region. That is because $f\left(\boldsymbol{x}\right)$ is convex and it assumes lower values at some $\boldsymbol{x} \in \Xi$ (see the level curves in Figure 44). However, in general, we could have the complete opposite effect, for example if $f\left(\boldsymbol{x}\right)$ is still convex but $\arg\min_{\boldsymbol{x}\in\Omega} f\left(\boldsymbol{x}\right) \not\subset \Xi$.

- Overall, all methods can perform quite well even when the global minimizers of the benchmark GOPs (2.1) are located on the boundary of Ξ (or close to it). That is the case for the `gramacy and lee constrained`, `sasena 1` [125], `townsend` [1], `camel six humps constrained` [156], `sasena 2` [125] and `step 2 constrained` benchmarks (at least, that we know of). As a matter of fact, in Table 6 and for BBO, at least one of the considered algorithms achieves a $d_{rel}\left(N_{max}\right) < 2\%$ on the aforementioned problems (except for the `step 2 constrained` benchmark).

- Similarly to what we have seen in Section 7.2, the empirical results presented in this Section can help us compare the performances of the BBO methods with those achieved by their PBO counterparts. In particular, if we consider only `C-GLIS-r` and `C-GLISp-r`, we can conclude that, in both frameworks, the methods attain similar results on several occasions (see $N_{acc>95\%}$ in Table 5 and $d_{rel}\left(N_{max}\right)$ in Table 6), although `C-GLIS-r` (BBO) is slightly more efficient than `C-GLISp-r` (PBO).

- Lastly, for what concerns the average execution times in Table 7, the proposed methods, i.e. `C-GLIS-r` and `C-GLISp-r`, are usually more computationally expensive than `C-GLIS` and `C-GLISp` [156] although, surprisingly, they are faster than the latter methods on some of the higher-dimensional benchmark problems.

Next, we compare `C-GLIS-r` PSVM with `C-GLIS-r` IDWI and `C-GLISp-r` PSVM with `C-GLISp-r` IDWI more in detail:

- If we consider only the convergence plots and the data profiles, we can see that, on several occasions, the robustness and efficiency of `C-GLIS-r` and `C-GLISp-r` are not particularly affected by the choice of the surrogate model for the probability of Ξ-feasibility. See for example the `gramacy and lee constrained`, `sasena 1` [125], `mishra's bird` [156] and `sasena 2` [125] benchmarks.

- *Overall, there is no clear winner between `C-GLIS-r` PSVM (respectively, `C-GLISp-r` PSVM) and `C-GLIS-r` IDWI (`C-GLISp-r` IDWI).* In some cases, the former can be more efficient, see for example the values of $N_{acc>95\%}$ for the `gramacy and lee constrained` problem (reported in Table 5). Instead, `C-GLIS-r` IDWI and `C-GLISp-r` IDWI can be slightly more robust, as highlighted by the data profiles for the `camel six humps constrained` [156] benchmark. For what concerns the higher-dimensional problems, `C-GLIS-r` PSVM performs better than

`C-GLIS-r` IDWI on the `himmelblau` [78] and `welded beam design` [78] benchmarks (mostly in BBO), while on the `step 2 constrained` problem the opposite is true.

- In practice, in the constrained framework, the robustness of an algorithm also depends on how fast it is able to find a Ξ-feasible candidate. In practice, when no Ξ-feasible samples are available, both `C-GLIS-r` and `C-GLISp-r` proceed by minimizing the IDW distance function $z_N(\boldsymbol{x})$ in (4.17) (pure exploration, see Section 6.2), without using the PSVM classifier nor the IDWI function in (4.16). Surprisingly, on the `camel six humps constrained` [156] benchmark, `C-GLIS-r` PSVM and `C-GLIS-r` IDWI, as well as `C-GLISp-r` PSVM and `C-GLISp-r` IDWI, require different amounts of samples to gain Ξ-feasibility on all N_{trial} trials. We attribute this behavior to the `PSWARM` [72] algorithm employed for the infill sampling criteria of the methods, which includes some form of randomization (see Section 1.2.5).

- Lastly, in Table 7, the two approaches show comparable execution times, although these are slightly more in favor of `C-GLIS-r` PSVM and `C-GLISp-r` PSVM.

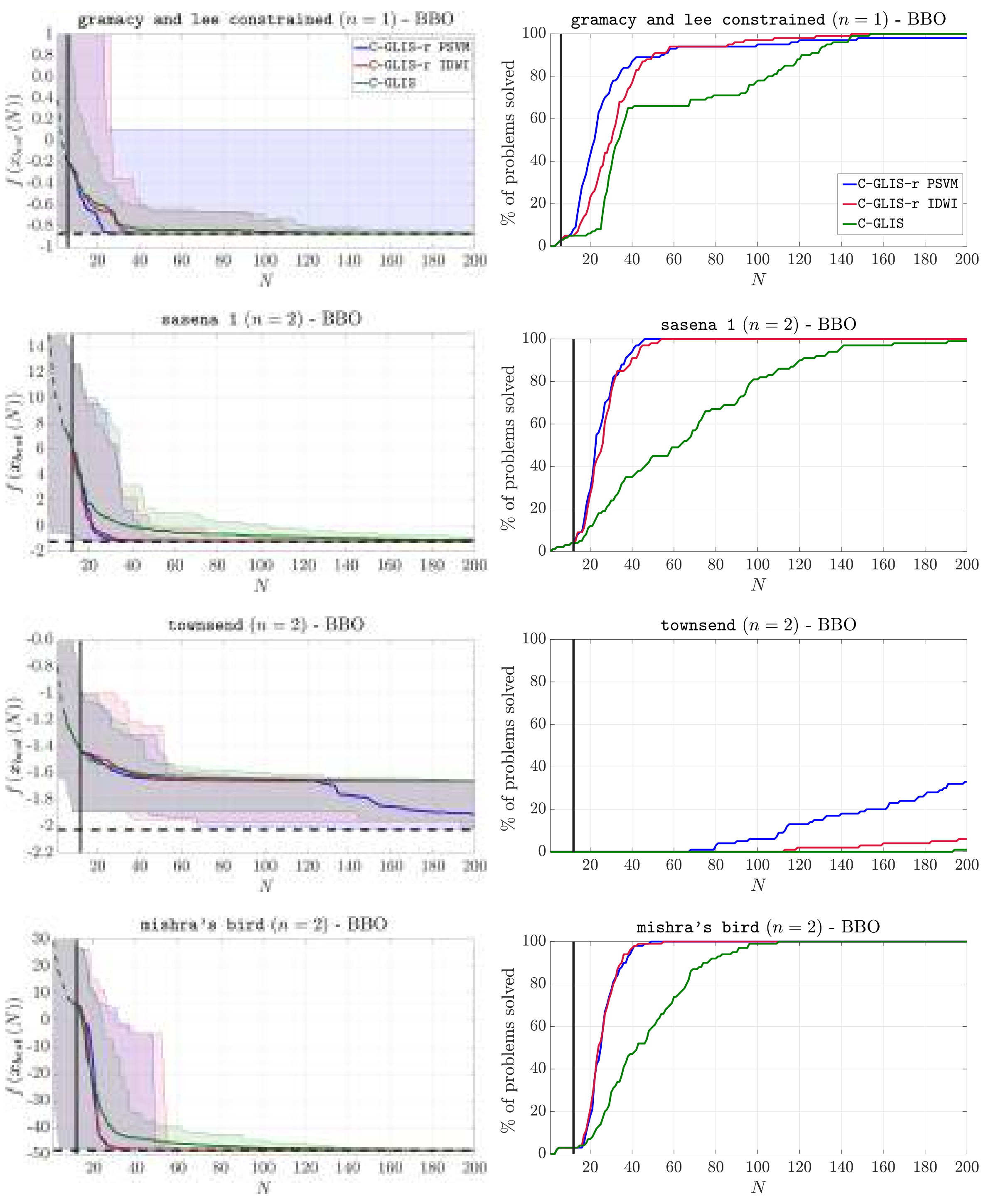


Figure 39: Performances achieved by the different constrained black-box optimization algorithms on the benchmark GOPs (2.1): convergence plots on the left and data profiles ($acc(N) > 95\%$) on the right. `C-GLIS-r` PSVM is depicted in blue, `C-GLIS-r` IDWI is shown in red and, lastly, `C-GLIS` in green. For what concerns the convergence plots, we use a dashed line for $f(\boldsymbol{x_{best}}(N))$ as long as the corresponding algorithm has not found a Ξ-feasible sample in all N_{trial} trials; when $\boldsymbol{x_{best}}(N) \in \Xi$ in all trials, we switch to a continuous line. The global minimum f^* is represented with a dashed black line. We also show the number of initial samples, N_{init}, with a black vertical line.

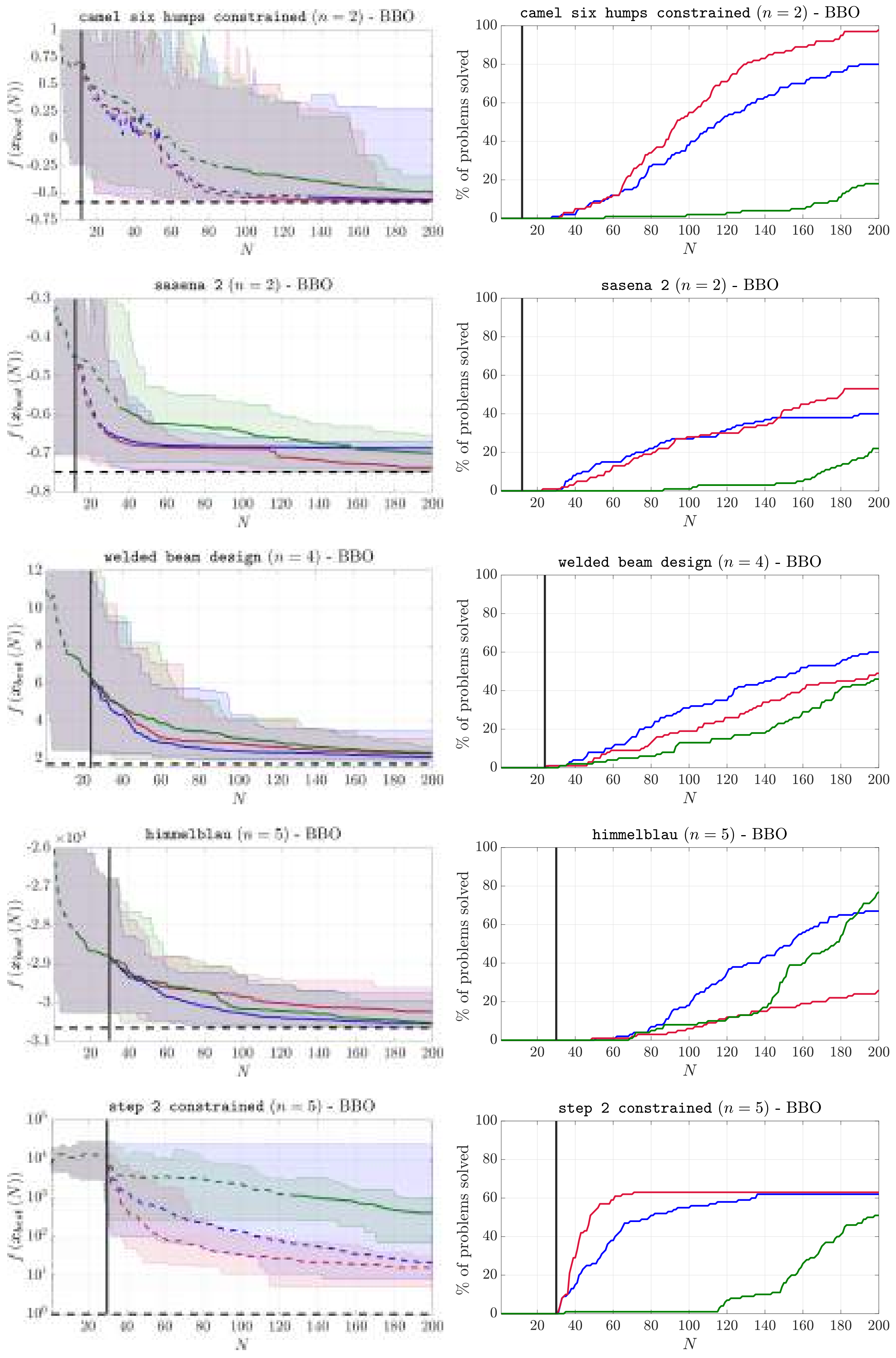


Figure 40: Figure 39 cont'd (constrained BBO).

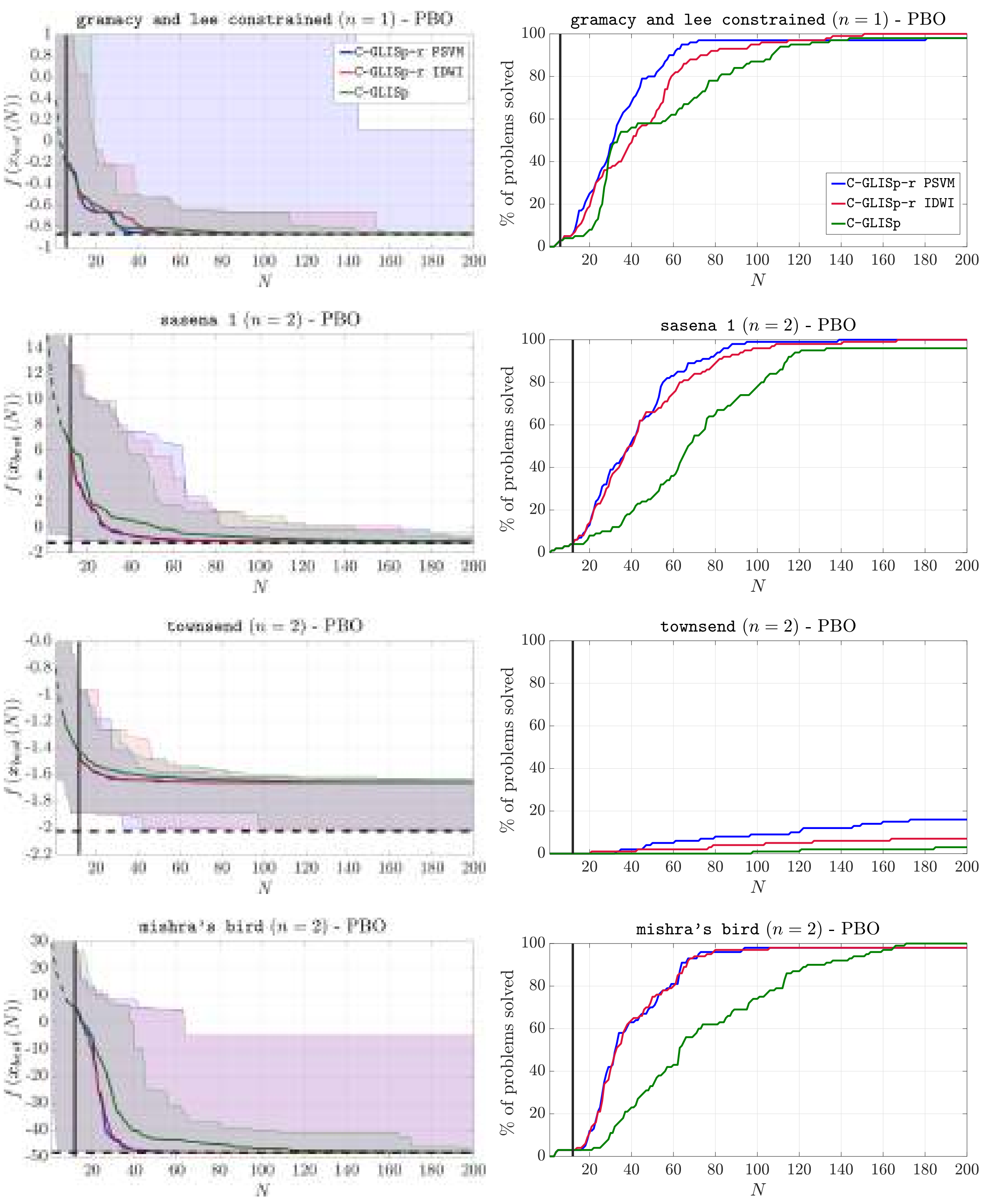


Figure 41: Performances achieved by the different constrained preference-based optimization algorithms on the benchmark GOPs (2.1): convergence plots on the left and data profiles ($acc(N) > 95\%$) on the right. `C-GLISp-r` PSVM is depicted in blue, `C-GLISp-r` IDWI is shown in red and, lastly, `C-GLISp` [156] in green. For what concerns the convergence plots, we use a dashed line for $f(x_{best}(N))$ as long as the corresponding algorithm has not found a Ξ-feasible sample in all N_{trial} trial; when $x_{best}(N) \in \Xi$ in all trials, we switch to a continuous line. The global minimum f^* is represented with a dashed black line. We also show the number of initial samples, N_{init}, with a black vertical line.

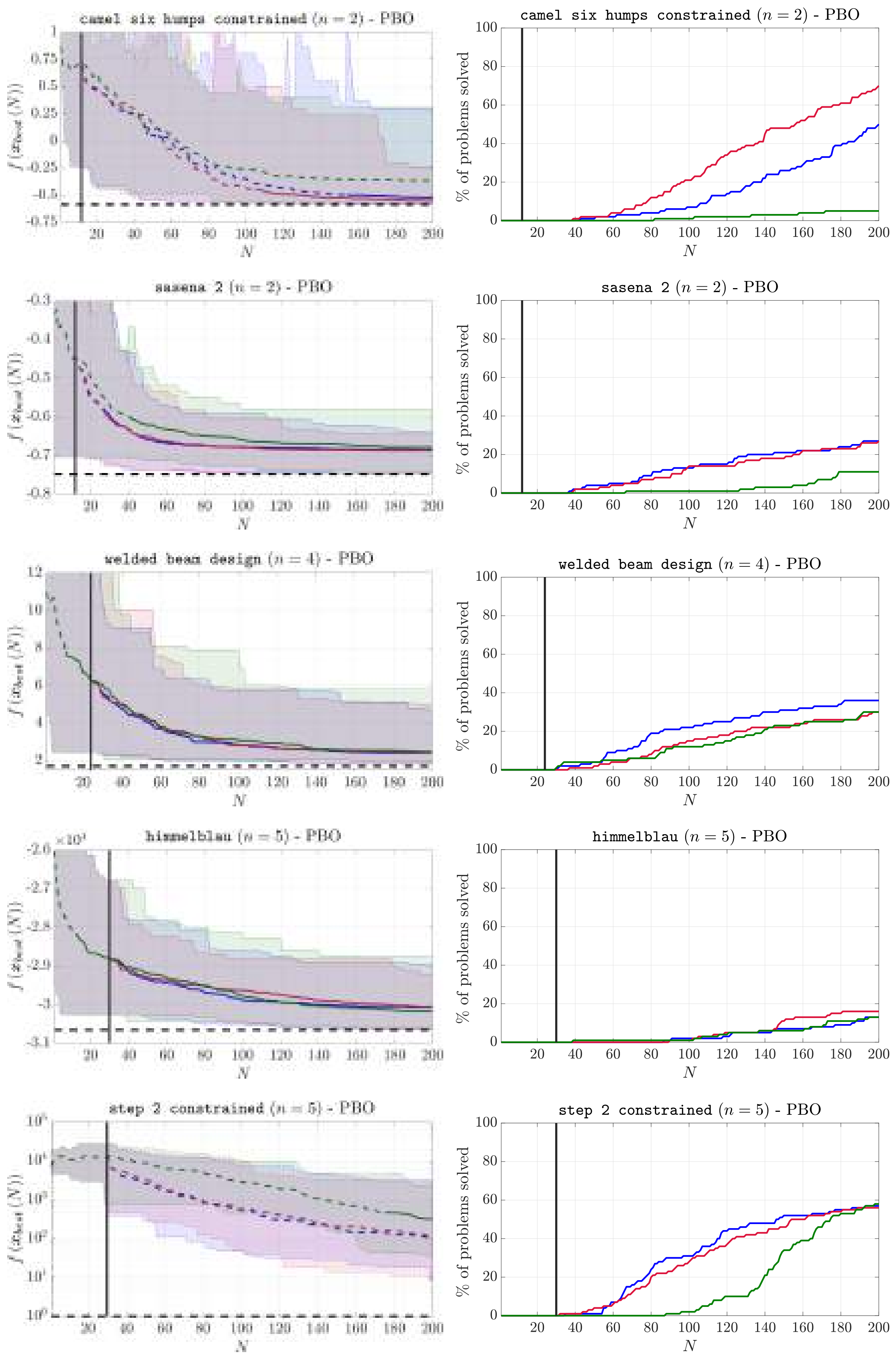


Figure 42: Figure 41 cont'd (constrained PBO).

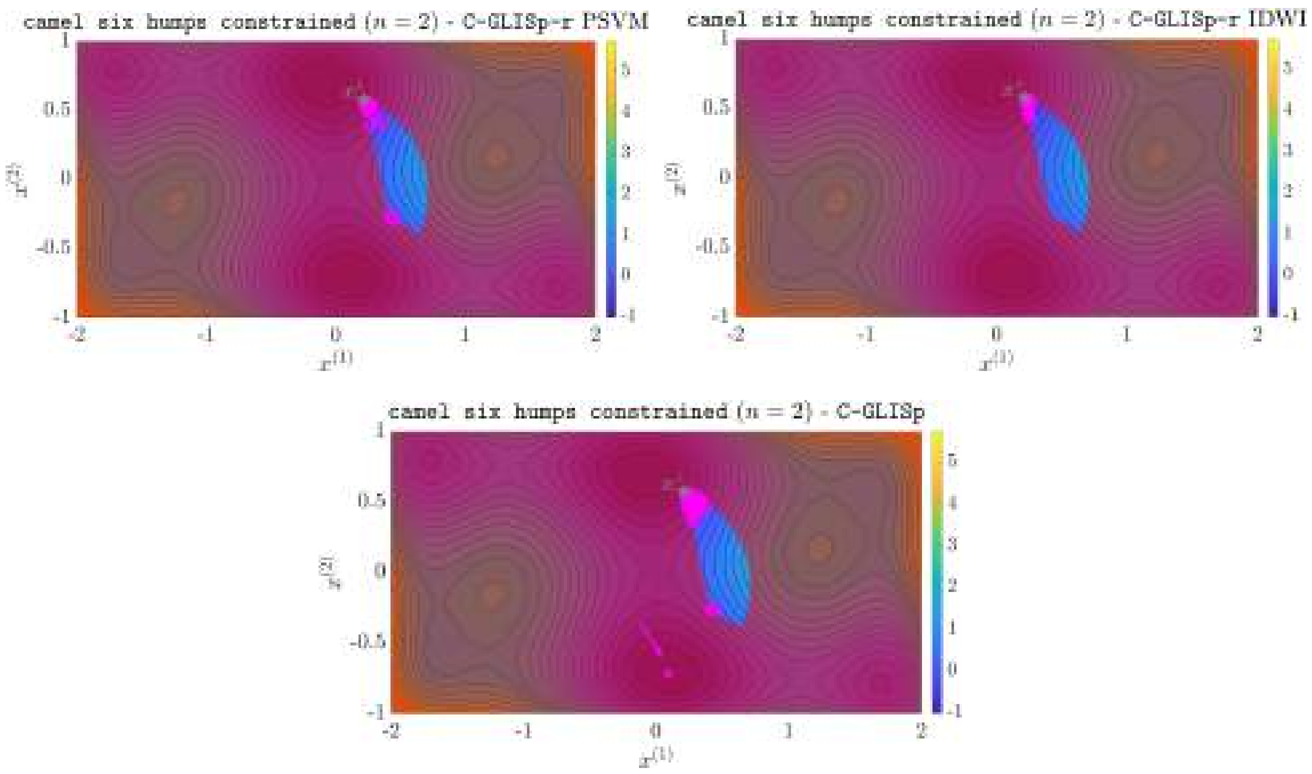
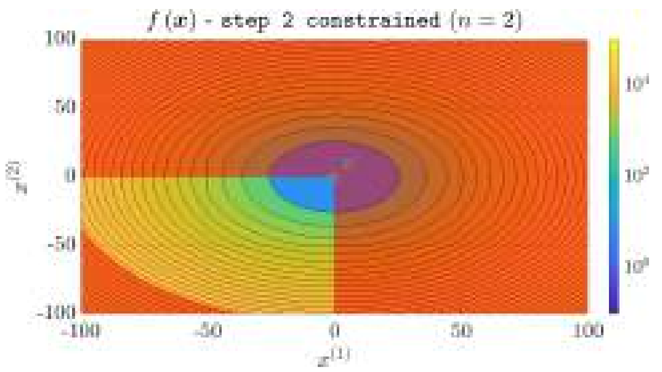


Figure 43: Best samples found by the constrained preference-based optimization procedures when solving the `camel six humps constrained` [156] benchmark. The shaded red area depicts the Ξ-infeasible region. Each point in magenta represents x_{best} (N_{max}) for the corresponding trial. We use a circle to highlight those samples that are Ξ-feasible and a cross for those that are not. The grey star depicts the global minimizer of the GOP (2.1). Notice how 2/100 trials performed by `C-GLISp` [156] return Ξ-infeasible samples, as highlighted by the arrow in magenta.



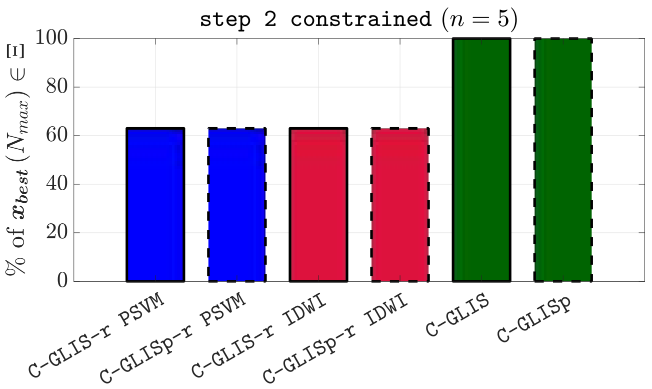


Figure 44: On the left: level curves of the cost function $f(x)$ of the `step 2 constrained` benchmark for $n = 2$ (instead of $n = 5$). The shaded red area denotes the Ξ-infeasible region. The global minimizer is shown as a grey star. On the right: number of Ξ-feasible solutions found by the BBO (continuous line) and PBO (dashed line) procedures on the actual `step 2 constrained` benchmark ($n = 5$).

Table 5: Number of samples required by the constrained black-box and preference-based optimization algorithms to solve the benchmark GOPs (2.1) **to a relative accuracy of** 95% **(median-wise). To take into account that black-box constraints are present, we start computing the relative accuracy only when $x_{best}(N) \in \Xi$ on all the trials of the corresponding algorithm. The acronym `n.r.` stands for "not reached". The best results are highlighted in bold font.**

			$N_{acc>95\%}$		
Benchmark problem	n	**Optimization**	`C-GLIS\p`	`C-GLIS\p-r` PSVM	`C-GLIS\p-r` IDWI
`gramacy and lee constrained`	1	black-box	34	**23**	31
		preference-based	34	**32**	41
`sasena 1` [125]	2	black-box	63	**23**	26
		preference-based	67	**39**	**39**
`townsend` [1]	2	black-box	`n.r.`	`n.r.`	`n.r.`
		preference-based	`n.r.`	`n.r.`	`n.r.`
`mishra's bird` [156]	2	black-box	43	25	**24**
		preference-based	63	**33**	**33**
`camel six humps constrained` [156]	2	black-box	`n.r.`	135	**101**
		preference-based	not all Ξ-feasible	200	**157**
`sasena 2` [125]	2	black-box	`n.r.`	`n.r.`	**180**
		preference-based	`n.r.`	`n.r.`	`n.r.`
`welded beam design` [78]	4	black-box	`n.r.`	**159**	`n.r.`
		preference-based	`n.r.`	`n.r.`	`n.r.`
`himmelblau` [78]	5	black-box	177	**155**	`n.r.`
		preference-based	`n.r.`	`n.r.`	`n.r.`
`step 2 constrained`	5	black-box	**194**	not all Ξ-feasible	not all Ξ-feasible
		preference-based	**175**	not all Ξ-feasible	not all Ξ-feasible

Table 6: Median relative distances from the global minimizer(s) of the considered benchmark GOPs (2.1) achieved by the constrained black-box and preference-based optimization algorithms. The best results are highlighted in bold font.

			$d_{rel}\,(N_{max})$		
Benchmark problem	n	**Optimization**	`C-GLIS\p`	`C-GLIS\p-r` PSVM	`C-GLIS\p-r` IDWI
`gramacy and lee constrained`	1	black-box	0.01%	**0.00%**	0.01%
		preference-based	0.04%	**0.00%**	0.01%
`sasena 1` [125]	2	black-box	0.84%	**0.09%**	0.13%
		preference-based	0.64%	**0.15%**	0.23%
`townsend` [1]	2	black-box	30.73%	**1.34%**	30.44%
		preference-based	**30.84%**	30.86%	30.87%
`mishra's bird` [156]	2	black-box	0.15%	**0.08%**	**0.08%**
		preference-based	0.14%	**0.04%**	0.05%
`camel six humps constrained` [156]	2	black-box	0.98%	0.28%	**0.13%**
		preference-based	not all Ξ-feasible	0.54%	**0.39%**
`sasena 2` [125]	2	black-box	3.16%	50.45%	**0.93%**
		preference-based	**5.72%**	50.50%	50.35%
`welded beam design` [78]	4	black-box	26.09%	**20.59%**	22.83%
		preference-based	30.43%	**26.23%**	30.76%
`himmelblau` [78]	5	black-box	**3.46%**	3.52%	9.62%
		preference-based	**20.11%**	20.88%	23.77%
`step 2 constrained`	5	black-box	**4.31%**	not all Ξ-feasible	not all Ξ-feasible
		preference-based	**3.82%**	not all Ξ-feasible	not all Ξ-feasible

Table 7: Average execution times of the constrained black-box and preference-based optimization algorithms. The best results are highlighted in bold font.

			Average execution times $[sec]$		
Benchmark problem	n	**Optimization**	`C-GLIS\p`	`C-GLIS\p-r PSVM`	`C-GLIS\p-r IDWI`
`gramacy and lee constrained`	1	black-box	**8.9**	65.8	63.9
		preference-based	**64.2**	120.5	116.0
`sasena 1` [125]	2	black-box	**35.5**	86.7	105.6
		preference-based	**88.4**	144.6	164.3
`townsend` [1]	2	black-box	**31.5**	92.0	117.9
		preference-based	**85.1**	157.9	168.7
`mishra's bird` [156]	2	black-box	**26.8**	82.9	98.1
		preference-based	**83.2**	143.0	169.4
`camel six humps constrained` [156]	2	black-box	**26.1**	101.0	110.5
		preference-based	**86.2**	156.0	173.3
`sasena 2` [125]	2	black-box	**29.4**	108.3	107.0
		preference-based	**86.1**	155.0	167.5
`welded beam design` [78]	4	black-box	137.3	**118.1**	145.4
		preference-based	208.4	**191.4**	222.9
`himmelblau` [78]	5	black-box	213.3	**121.5**	149.8
		preference-based	310.1	**218.2**	257.1
`step 2 constrained`	5	black-box	**62.7**	212.8	221.2
		preference-based	**160.4**	259.6	283.8

7.4 Chapter summary

In this Chapter, we have thoroughly compared the original methods, `GLIS` [10], `GLISp` [11], `C-GLIS` and `C-GLISp` [156], with the proposed extensions, `GLIS-r` [108], `GLISp-r` [109], `C-GLIS-r` and `C-GLISp-r`.

In the unconstrained preference-based optimization framework, algorithm `GLISp-r` [109] is more robust than `GLISp` [11] (i.e. it is able to locate the global minima of the GOP (2.1) more frequently, regardless of the starting points) due to the proposed modifications, namely the min-max rescaling of the terms of the acquisition function in (5.12) and the greedy δ-cycling strategy (in Section 5.2.3). Instead, `GLISp` [11] is often able to find a good solution slightly faster than the proposed extension, although it might just be a local one. In practice, the latter method behaves quite similarly to a pure exploitatory ("greedy") procedure. As matter of fact, `GLISp` [11] is comparable to `GLISp-r` [109] when equipped with $\Delta_{cycle} = \langle 0.95 \rangle$ ("pure" exploitation). Instead, in the unconstrained black-box optimization framework, `GLIS` [10] and `GLIS-r` [108] exhibit similar performances on several occasions. Due to the presence of the IDW variance function $s_N(\boldsymbol{x})$ in (4.19), the original method does not appear so prone to getting stuck on local minima of the GOP (2.1) (as it happens for `GLISp` [11], in PBO), surely less so than if only the IDW distance function $z_N(\boldsymbol{x})$ in (4.17) were to be the sole exploratory contribution for the acquisition function $a_N(\boldsymbol{x})$ in (4.20a) (e.g. if $\delta_2 = 0$). Lastly, `C-GLIS` and `C-GLISp` [156] are often as robust as `GLIS-r` [108] and `GLISp-r` [109] but notably slower.

In the constrained black-box and preference-based optimization frameworks, `C-GLIS-r` and `C-GLISp-r`, equipped with either the revisited PSVM classifier in Section 6.1.3 or the IDWI function in (4.16), are remarkably more efficient than `C-GLIS` and `C-GLISp` [156] on most of the considered benchmark GOPs (2.1) (especially the lower-dimensional ones) while being at least as robust as the latter methods. The only exception is the `step 2 constrained` benchmark which, due to its definition, favors the infill sampling criteria of `C-GLIS` and `C-GLISp` [156]. For what concerns `C-GLIS-r` and `C-GLISp-r`, we have shown that both the revisited PSVM classifier in Section 6.1.3 and the IDWI function in (4.16) are viable surrogates for the probability of Ξ-feasibility.

The only disadvantage of the proposed procedures is the increased computational time, which is due to the computational overhead of Algorithm 13 (used for the generation of the augmented sample set $\mathcal{X}_{aug}$). However, we argue that the observed computational overhead is negligible when compared to the time spent on performing simulations or experiments, i.e. when the surrogate-based procedures are actually applied to real problems (see Assumption 2.3), as we will see in the next Chapter.

As a final remark, the empirical results reported in this Chapter are also particularly useful for comparing BBO and PBO procedures. Our results show that preference-based optimization methods are able to achieve similar (or slightly worse) performances to those obtained by their black-box counterparts, despite using less information on the cost function $f(\boldsymbol{x})$ of the GOP (2.1).

Chapter 8. Case study: calibration of the position controller of a hydraulic forming press

This Chapter is devoted to the application of the proposed black-box and preference-based optimization algorithms to a control systems case study. We consider the task of *calibrating the position controller of a hydraulic forming press*. Consistently with the surrogate-based method rationale in Section 2.2 (Figure 4), we have at our disposal a simulator of the system under study, which we use to tune the regulator's parameters.
The remainder of this Chapter is organized as follows. In Section 8.1, we give a brief introduction of the control systems application at hand. After that, the hydraulic press under study is described in detail in Section 8.2. Then, in Section 8.3, we formalize the control specifications and translate them into suitable cost functions and black-box constraints that can be used for black-box optimization of the regulator's parameters. For what concerns preference-based optimization, the author of this book plays the role of the calibrator, who states preferences and assesses which tunings are acceptable and which are not (decision-maker-based constraint, see Definition 3.6). The performances of `GLIS-r` [108], `GLISp-r` [109], `C-GLIS-r` and `C-GLISp-r` are analyzed in detail in Section 8.4. Lastly, Section 8.5 is devoted to some concluding remarks.

8.1 Introduction and motivation

Press forming [63, 105, 140] is a mechanical process wherein a material undergoes deformations to achieve a desired shape. The press forming process can be summarized as follows (see Figure 45):

1. A sheet of plastic or metallic material is clamped on a die through blank holders;
2. A punch is attached to the mobile plane of the forming press, directly above the die;
3. The mobile plane is lowered towards the sheet of material in a controlled manner;
4. Once the punch reaches the work-piece, a force is applied to it and the sheet of material is deformed to achieve the desired shape;
5. The mobile plane of the forming press is raised, separating the punch from the die;
6. The work-piece is extracted.

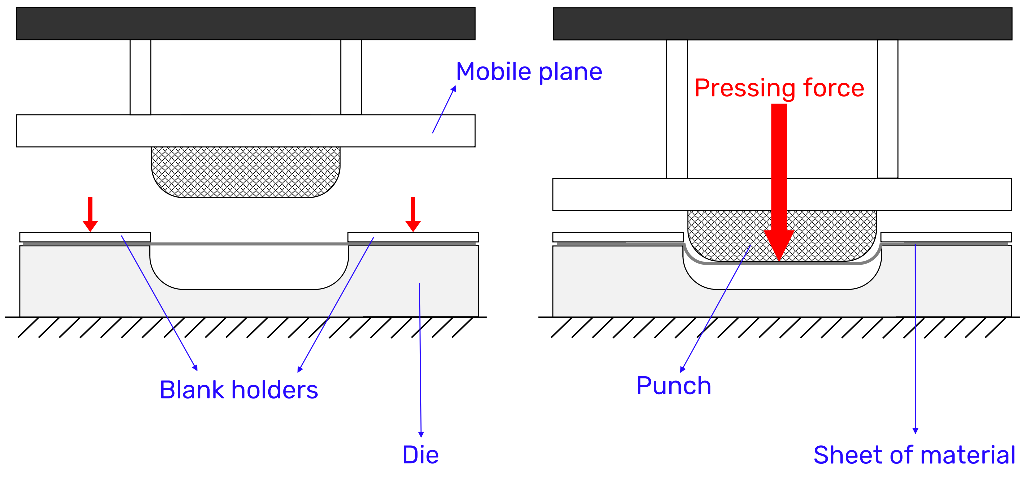


Figure 45: The press forming process and its components.

Hydraulic presses are widely used in heavy-duty forming processes due to their high power-to-mass ratios, high stiffnesses and high load capabilities [82, 133]. The pressing force is generated by a pressurized fluid (usually oil) that causes the motion of one or several piston rods, which are mechanically connected to the mobile plane of the press. The fluid is fed to the hydraulic cylinders, which house the pistons, through pipes. The flow rates of the fluid going to the hydraulic cylinders are regulated by electromechanical valves.

Typically, hydraulic presses are controlled in closed-loop and in two phases [133]: the lowering and raising of the mobile plane are regulated by a *position controller*, while the pressing of the work-piece (which starts once the punch reaches it) is handled by a *force controller*. The most commonly used controllers for both tasks are Proportional-Integral-Derivative (PID) regulators [4], due to their simplicity and reliability. Suitable position and force profiles are defined based on the press forming operation at hand. In any case, the regulators must be tuned so that several *control specifications* are met, most importantly: (i) the mobile plane of the hydraulic press must not show any oscillations when lowered or raised, (ii) the tracking of the position profile must be as accurate as possible, especially when the punch is close to the work-piece, and (iii) in the pressing phase, during which the force controller is active, overshoots must be limited to avoid damaging the sheet of material.

Traditional controller tuning strategies are model-based [43, 102]: the regulator's parameters are computed from a model of the system under study, following ad hoc tuning rules (see for example [101] for PID controllers). Most calibration criteria in the model-based setting are designed for linear systems and are the result of the optimization of some performance indicators (such as the ones reviewed in [36]). However, hydraulic presses are complex systems which can exhibit nonsmooth and discontinuous nonlinearities [88, 140]. These nonlinearities can result, for example, from the nonlinear relationship between the pressure inside the hydraulic cylinder's chambers and the fluid flow rate, the valve opening profiles (which can exhibit a dead-band) and friction. Physics-based modeling of hydraulic presses is no easy task; even after deriving a model from first principles, many of its parameters are unknown and must be estimated. These include, for example, the friction coefficients, the parameters of the transfer functions which describe the electromechanical valves and the bulk modulus of the fluid. To estimate the model parameters, we must carry out suitable open-loop and/or closed-loop experiments on the hydraulic press and employ an identification procedure [86, 145]. In practice, the identification-oriented experiments[1] can be quite time-consuming or even impracticable (e.g. if they involve an input signal that puts too much stress on the actuators). We also point out that, even after we successfully estimate the model parameters and tune the controller of the hydraulic press by means of a proper tuning rule, the resulting calibration might be subject to some (manual) fine-tuning to better meet the control specifications. In practice, if the system is highly nonlinear, then it is quite unlikely that there already exist off-the-shelf calibration criteria suited for it.

It is common in industrial practice to forego the hydraulic press modeling completely due to its complexity. Instead of relying on a model-based tuning approach, a suitable controller tuning is sought by an experienced calibrator (decision-maker), who performs several closed-loop experiments on the system following a trial-and-error approach. *This is where black-box and preference-based optimization procedures shine* (see Example 2.1 and Example 3.4). Instead of letting the DM choose which tunings to try next, we guide him/her during the experiments by proposing new calibrations, following the surrogate-based method rationale in Section 2.2. If the control specifications are well-defined, we can derive a suitable cost function and employ a BBO procedure to drive the search; otherwise, we can guide the calibrator based on his/her preferences (PBO). Compared to the trial-and-error methodology, BBO and PBO algorithms are more structured and reduce the number of experiments required to obtain an adequate controller calibration. Furthermore, differently from the model-based approach, surrogate-based methods do not require a model of the hydraulic press beforehand. Lastly, there is no need to run any identification-oriented experiment. Instead, the

[1]Most often, the identification-oriented experiments are quite different from the experiments used to assess the performances of a calibration. The former involve input signals that are able to "excite" the system under study "well enough" to produce output signals that are meaningful for the identification of its model (see [86, 145]). Instead, the latter only use input signals that are suited for the task at hand. To clarify this concept, consider the hydraulic press described in this Section and controlled in closed-loop. During the lowering phase, the input signal is the position reference signal that the mobile plane of the press must follow. In a performance-oriented experiment, the input signal is selected as the position profile that is commonly used during the material forming operation. Instead, good input signals for identification-oriented experiments are white noise signals, pseudorandom binary sequences and multisines, which differ completely from the signals used during normal working operation of the hydraulic press.

experiments required to perform sample evaluations (i.e. to assess the performances of a given controller calibration) are carried out in the same fashion as during normal working operation of the hydraulic press.

8.2 System description

We consider the hydraulic press depicted in Figure 46. The *mobile plane* of the press is mechanically connected to the *piston rods* of four *hydraulic cylinders*, which cause its motion. The inner cylinders are single-acting whereas the outer ones are double-acting[2]. The force produced by each hydraulic cylinder depends on the pressure inside its chambers and on its constructive parameters (such as the area of the piston). The pressure inside the cylinders' chambers is generated by *pressurized oil*, which flows to the cylinders through several different pipes. The flow rates of the oil inside each pipe are regulated by different electromechanical valves. The *filling valves* regulate the oil throughput from the tank to the upper chambers of the hydraulic cylinders. These valves are either completely open or completely closed and can handle high flow rates. Instead, the *pressing valves* and the *balance valves* allow for a continuous regulation (between 0% and 100%) of the oil throughput but possess lower nominal flow rates. In particular, the positions of the valves' members, which obstruct the oil flow, are controlled in closed-loop (using integrated PID controllers). In the system under study, the pressing valves regulate the fluid flow rates to the upper chambers of the single-acting and double-acting hydraulic cylinders, whereas the balance valves handle the lower chambers of the double-acting hydraulic cylinders.
The working cycle of the hydraulic press in Figure 46 consists of several different phases:

1. The *waiting phase*, during which the mobile plane of the press is kept stable at a certain height, waiting for the start command.

2. The *closing phase*, wherein the mobile plane is lowered towards the work-piece and must follow a specific position profile. We can distinguish three stages: at first, we have the *fast descent* phase, followed by the *slow approach* and the *compression* phases. The former stage adopts a third-degree polynomial as the position profile to quickly approach the work-piece. Instead, the latter two phases use a first-degree polynomial as the position profile and start when the mobile plane is very close to the sheet of material.

[2]In a single-acting cylinder, the pressurized fluid extends the piston rod only in one direction. Then, its retraction is caused either by an inbuilt spring or due to gravity. Instead, in a double-acting cylinder, both the extension and the retraction of the piston rod are driven by the pressurized fluid.

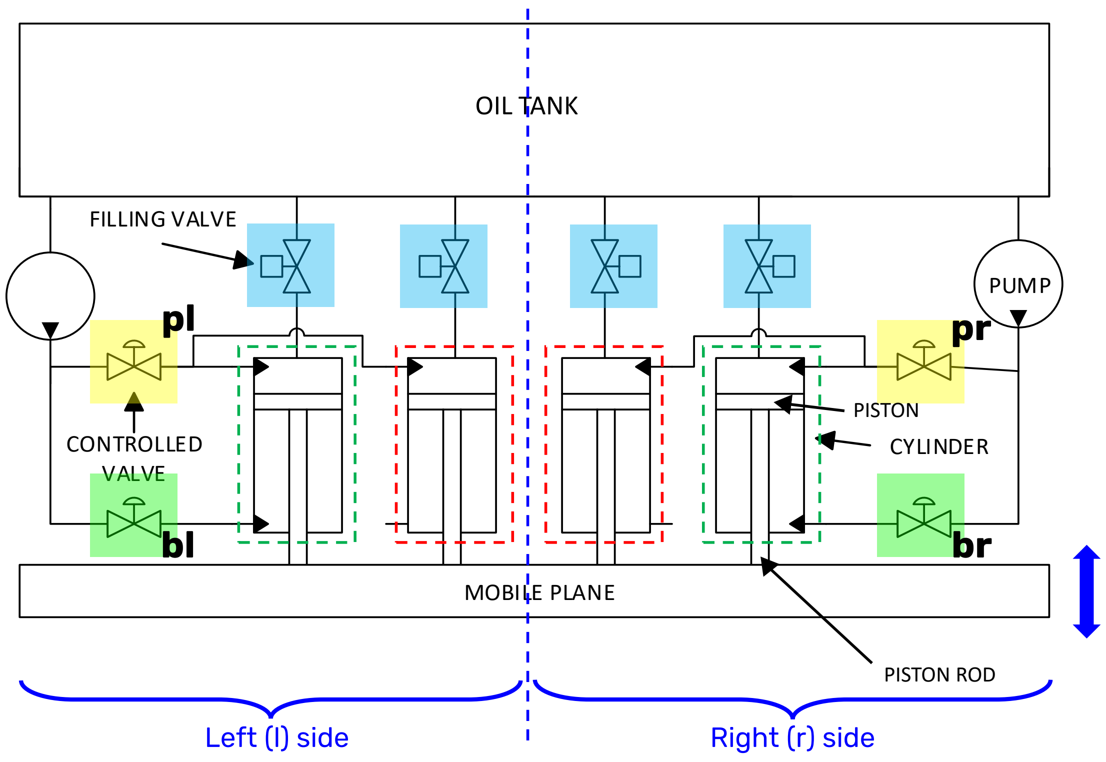


Figure 46: Simplified hydraulic scheme of the system under study. We distinguish between the left and the right side of the press. The cylinders highlighted by a dashed green rectangle are double-acting, whereas the ones contained inside a dashed red rectangle are single-acting. Several different valves are present: filling valves (blue), pressing valves (yellow) and balance valves (green). The acronyms pl, pr, bl and br stand for pressing left, pressing right, balance left and balance right respectively.

3. The *pressing phase*, which is tasked with deforming the work-piece. In this stage, the mobile plane must exert a proper force in order to correctly shape the sheet of material.

4. The *opening phase*, wherein the mobile plane of the press is raised to allow the extraction of the formed work-piece.

Figure 47 depicts a possible position profile of the closing phase for the hydraulic press under study. The opening phase typically follows a position profile that is specular to the one used in the closing phase. During the opening and closing phases, the position of the mobile plane of the press is regulated in closed-loop. In particular, we have a total of four PI controllers, one for each balance and pressing valve, that regulate the valves' opening percentages. The regulators of the balance valves can be equipped with an additional Feed Forward (FF) action, which can make the controllers more responsive. In the pressing phase, the PI (+ FF) controllers regulate the force exerted by the mobile plane instead of its position. *The tunings of the regulators vary at each stage since different control specifications must be met*. In Section 8.3, we will cover the control specifications and the calibration strategies for the PI (+ FF) controllers used during the fast descent phase of the hydraulic press.

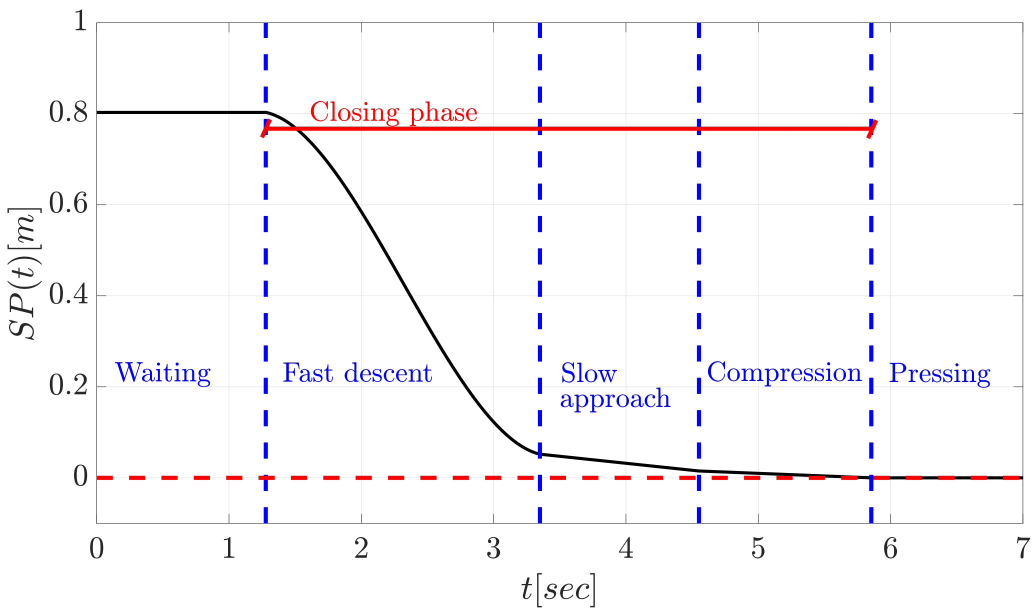


Figure 47: Example of position profile of the closing phase for the hydraulic press under study. We highlight the different stages: fast descent, slow approach and compression. We also include the waiting and pressing phases, although the position controller is not active during such stages. The dashed red line represents the position of the sheet of material with respect to the mobile plane of the press.

The *position control scheme* of the considered hydraulic press is shown in Figure 48. Several variables and sub-systems are present:

- We use the variable t to denote the time for the time-domain signals of interest (see Example 2.1);
- $SP\left(t\right)$ is the position reference signal (or *setpoint*), such as the one depicted in Figure 47;
- $pos_{(\cdot)}\left(t\right)$ are the positions of the left and right sides of the mobile plane of the press, which constitute the *controlled variables*. We use the subscript $(\cdot)$ to denote either the left, l, or the right, r, side of the press. Furthermore, we also define the corresponding velocities and accelerations as $vel_{(\cdot)}\left(t\right) = \frac{d}{dt}pos_{(\cdot)}\left(t\right)$ and $acc_{(\cdot)}\left(t\right) = \frac{d}{dt}vel_{(\cdot)}\left(t\right)$ respectively.
- $err_{(\cdot)} = SP\left(t\right) - pos_{(\cdot)}\left(t\right)$ are the position tracking errors;
- Each PI controller has two parameters: the proportional gain $K_{P_{(\circ\cdot)}}$ and the integral gain $K_{I_{(\circ\cdot)}}$. In the subscript $(\circ\cdot)$, $\circ$ is either p (pressing valve) or b (balance valve), while $\cdot$ is either l (left) or r (right).
- Each FF action depends only on one parameter, the feed forward gain, namely $K_{FF_{bl}}$ and $K_{FF_{br}}$ for the left and right balance valves respectively;

- The *control actions* generated by the PI (+ FF) regulators are the opening percentages for each electromechanical valve, i.e. $op_{(\circ\cdot)}(t)$;
- $fr_{(\circ\cdot)}(t)$ are the flow rates of the pressurized oil inside the different pipes after the fluid has been obstructed by the balance and pressing valves' members. Note that the fluid coming from the pressing valves is split between the upper chambers of the single-acting and double-acting hydraulic cylinders by the distributors. Instead, the flow rates of the pressurized oil coming from the four filling valves are denoted as $fr_{f,1}(t), \ldots, fr_{f,4}(t)$.
- The pressures inside the hydraulic cylinders' chambers, generated by the pressurized oil coming from the different valves, cause the pistons' rods to extend (or retract), exerting a force on the mobile plane of the hydraulic press. We denote the force produced by each hydraulic cylinder as $for_1(t), \ldots, for_4(t)$.
- Lastly, the forces generated by the hydraulic cylinders set the mobile plane in motion. The motion is affected by friction and depends on the inertia of all the components that compose the hydraulic forming press.

Formally, each signal, except for the setpoint $SP(t)$, should also depend on the controllers' parameters and not only on the time t, as we will see in Section 8.3 and Section 8.4.

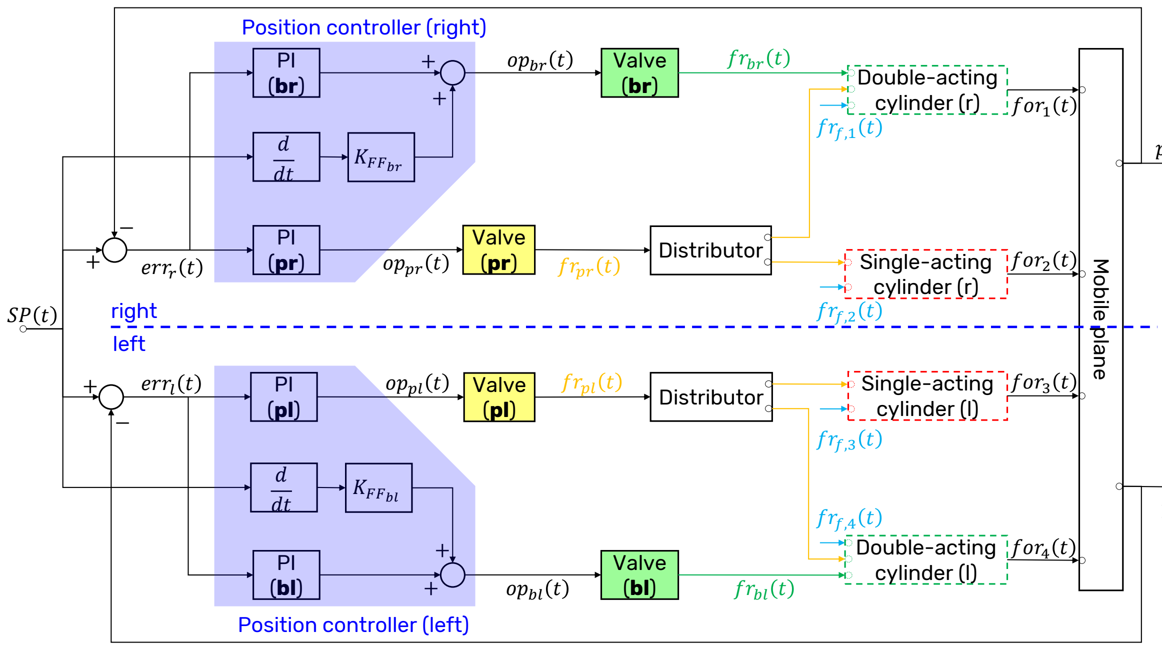


Figure 48: Position control scheme of the system under study. The colors follow the same conventions of Figure 46.

We have at our disposal a high-fidelity simulator of the hydraulic press depicted in Figure 46 and controlled as in Figure 48. A model of the system under study has been derived by combining well-known physics laws [26, 55, 88] with the data provided in the datasheets of each component. The unknown model parameters have been estimated by minimizing the deviation (in a Least Squares [57] sense) between the signals simulated from the derived model and the signals coming from several experiments on the real system.

8.3 Control specifications and calibration strategies

In this case study, we have focused on *tuning the PI + FF controllers of the balance valves for the fast descent phase of the hydraulic press*, which is the most demanding stage performance-wise. Throughout the fast descent phase, the pressing valves are left closed[3] (i.e. they are not controlled); therefore, we only need to calibrate the regulators of the balance valves. Consistently with industrial practice, we assume the hydraulic press to be symmetric. Therefore, we use the same calibration for both PI + FF controllers of the balance valves, i.e.:

$$\begin{aligned}
&K_P = K_{P_{br}} = K_{P_{bl}},\\
&K_I = K_{I_{br}} = K_{I_{bl}},\\
&K_{FF} = K_{FF_{br}} = K_{FF_{bl}}.
\end{aligned}$$

Hence, the decision vector simply amounts to:

$$\boldsymbol{x} = \begin{bmatrix} K_P & K_I & K_{FF} \end{bmatrix}^\top, \quad \boldsymbol{x} \in \mathbb{R}^3_{\geq 0}. \tag{8.1}$$

From now on, to highlight that the signals in Figure 48 depend on the controller calibration, we use the notation $pos_l\left(t;\boldsymbol{x}\right)$ for the position of the left side of the press and similarly for all the other signals (except for the setpoint $SP\left(t\right)$, which does not depend on $\boldsymbol{x}$).
The *control specifications* for the fast descent phase of the hydraulic press under study are:

1. The position trajectory tracking must be "good enough";
2. The position tracking errors, $err_{(\cdot)}\left(t;\boldsymbol{x}\right)$, cannot be "too big";
3. The downward motion of the mobile plane of the hydraulic press must be "as smooth as possible".

In practice, the available control specifications only give us qualitative information on the desired performances and must be translated into suitable indicators that can be employed for black-box

[3]As the name implies, the pressing valves are actuated only during the pressing phase. That is because whenever the oil is let through the pressing valves, the pressures inside the upper chambers of the cylinders increase and thus the mobile plane can exert a higher force, easing the deformation of the work-piece.

optimization. We propose three indicators, one for each of the aforementioned control specifications, and compute them from the signals produced by the simulator of the hydraulic press. Each signal is sampled at a sampling time $T_s \in \mathbb{R}_{>0}$, obtaining a total of $T \in \mathbb{N}$ samples. The proposed indicators are:

1. The trajectory tracking performances associated to a calibration $\boldsymbol{x} \in \mathbb{R}^3_{\geq 0}$ are described by the *position normalized mean absolute error*, $pos\_nmae_{(\cdot)} : \mathbb{R}^3_{\geq 0} \to \mathbb{R}_{\geq 0}$, defined as:

$$\begin{aligned} pos\_nmae_{(\cdot)}\left(\boldsymbol{x}\right) &= \frac{\sum_{t=1}^{T} \left|SP(t) - pos_{(\cdot)}(t;\boldsymbol{x})\right|}{\sum_{t=1}^{T} \left|SP(t) - \text{avg}_t\left[SP(t)\right]\right|} \cdot 100 \\ &= \frac{\sum_{t=1}^{T} \left|err_{(\cdot)}(t;\boldsymbol{x})\right|}{\sum_{t=1}^{T} \left|SP(t) - \text{avg}_t\left[SP(t)\right]\right|} \cdot 100, \end{aligned} \tag{8.2}$$

   where

$$\text{avg}_t\left[SP(t)\right] = \frac{1}{T} \cdot \sum_{t=1}^{T} SP(t)$$

   is the sample mean of the setpoint signal. $pos\_nmae_{(\cdot)}\left(\boldsymbol{x}\right) = 0$ if and only if the calibration $\boldsymbol{x}$ achieves perfect tracking throughout the whole fast descent phase (i.e. $pos_{(\cdot)}\left(t;\boldsymbol{x}\right) = SP(t), \forall t = 1,\ldots,T$), otherwise $pos\_nmae_{(\cdot)}\left(\boldsymbol{x}\right) > 0$.

2. The *maximum position tracking error* obtained by a tuning $\boldsymbol{x} \in \mathbb{R}^3_{\geq 0}$, $err\_max_{(\cdot)} : \mathbb{R}^3_{\geq 0} \to \mathbb{R}_{\geq 0}$, is:

$$\begin{aligned} err\_max_{(\cdot)}\left(\boldsymbol{x}\right) &= \max_{t \in \{1,\ldots,T\}} \left|SP(t) - pos_{(\cdot)}(t;\boldsymbol{x})\right| \\ &= \max_{t \in \{1,\ldots,T\}} \left|err_{(\cdot)}(t;\boldsymbol{x})\right|. \end{aligned} \tag{8.3}$$

   Clearly, a good controller must exhibit a low $err\_max_{(\cdot)}\left(\boldsymbol{x}\right)$. Moreover, similarly to $pos\_nmae_{(\cdot)}\left(\boldsymbol{x}\right)$ in (8.2), if $err\_max_{(\cdot)}\left(\boldsymbol{x}\right) = 0$, then the controller with calibration $\boldsymbol{x}$ achieves perfect tracking.

3. Smooth descent of the mobile plane of the hydraulic press results from smooth velocity signals, i.e. $vel_{(\cdot)}(t;\boldsymbol{x})$ must not show pronounced oscillations or abrupt changes. This means that their derivatives, i.e. the accelerations $acc_{(\cdot)}(t;\boldsymbol{x})$, must exhibit a small variance. Thus, one way to quantify the smoothness of the descent of the mobile plane is by computing the *standard deviation of the acceleration signals*, $acc\_std_{(\cdot)} : \mathbb{R}^3_{\geq 0} \to \mathbb{R}_{\geq 0}$, namely:

$$acc\_std_{(\cdot)}\left(\boldsymbol{x}\right) = \text{std}_t\left[acc_{(\cdot)}(t;\boldsymbol{x})\right], \tag{8.4}$$

   where

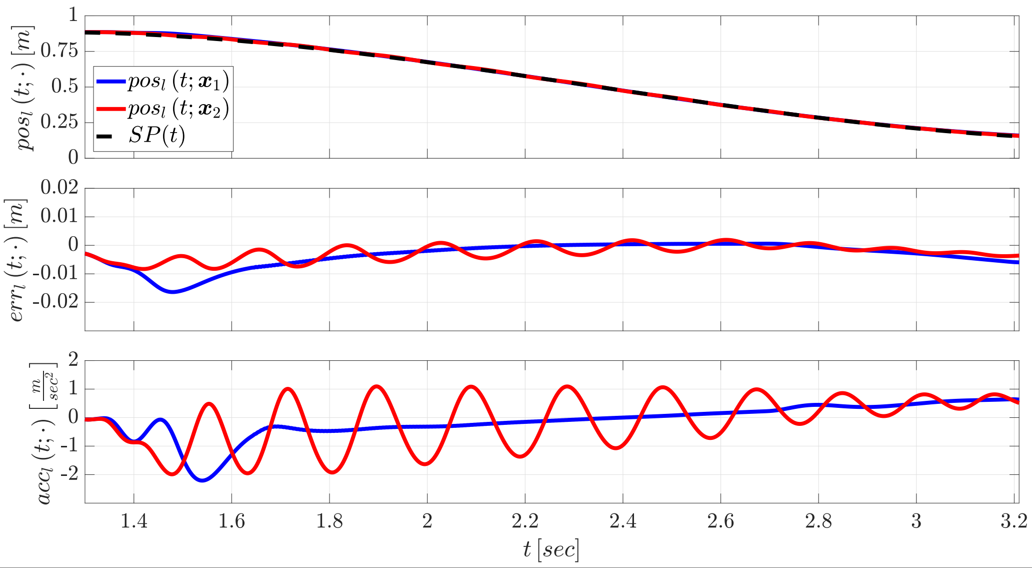
$$\text{std}_t\left[acc_{(\cdot)}(t;\boldsymbol{x})\right] = \sqrt{\frac{1}{T-1} \cdot \sum_{t=1}^{T} \left\{acc_{(\cdot)}\left(t;\boldsymbol{x}\right) - \text{avg}_t\left[acc_{(\cdot)}(t;\boldsymbol{x})\right]\right\}^2}$$

is the sample standard deviation of $acc_{(\cdot)}(t;\boldsymbol{x})$. In practice, linear trends resulting from the specific position profile $SP(t)$ must be removed from $acc_{(\cdot)}(t;\boldsymbol{x})$ before computing $acc\_std_{(\cdot)}(\boldsymbol{x})$ in (8.4).

The proposed performance indicators are averaged between the left and right sides of the press. In particular, we define:

$$pos\_nmae(\boldsymbol{x}) = \frac{pos\_nmae_l(\boldsymbol{x}) + pos\_nmae_r(\boldsymbol{x})}{2}$$

and similarly for $err\_max_{(\cdot)}(\boldsymbol{x})$ in (8.3) and $acc\_std_{(\cdot)}(\boldsymbol{x})$ in (8.4). As an example, Figure 49 shows the performances achieved by two different calibrations with respect to the proposed indicators.



Calibration $\boldsymbol{x}_i$	K_P	K_I	K_{FF}	$pos\_nmae(\boldsymbol{x}_i)$	$err\_max(\boldsymbol{x}_i)\ [m]$	$acc\_std(\boldsymbol{x}_i)\ \left[\frac{m}{sec^2}\right]$
$\boldsymbol{x}_1$ (blue)	0.736	2.154	0.046	1.689	0.016	0.350
$\boldsymbol{x}_2$ (red)	2.000	5.738	0.023	1.323	0.008	0.742

Figure 49: Example of position, position tracking error and acceleration signals achieved by two different calibrations, $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$, of the PI + FF controllers. Only the signals related to the left side of the hydraulic press are shown. Tuning $\boldsymbol{x}_2$ shows better tracking performances (lower $pos\_nmae(\boldsymbol{x})$ and $err\_max(\boldsymbol{x})$) but worse accelerations (higher $acc\_std(\boldsymbol{x})$). Vice-versa for the calibration $\boldsymbol{x}_1$.

8.3.1 Calibration strategies

We employ algorithms `GLIS-r` [108], `GLISp-r` [109], `C-GLIS-r` and `C-GLISp-r` to calibrate the parameters of the controllers. In particular, we consider the global optimization problem in (2.1), namely:

$$\mathcal{X}^* = \arg\min_{\boldsymbol{x}} f(\boldsymbol{x})$$
$$\text{s.t.} \quad \boldsymbol{x} \in \Omega \cap \Xi,$$

where Ω is composed of simple bounds on the controllers' parameters:

$$\begin{aligned} 0.1 \leq & K_P \leq 2, \\ 0.1 \leq & K_I \leq 10, \\ 0.01 \leq & K_{FF} \leq 0.1, \end{aligned} \tag{8.5}$$

while $f(\boldsymbol{x})$ and Ξ will be defined shortly. The bounds in (8.5) have been derived from the calibrations used by several other hydraulic presses, which adopt the same position control scheme in Figure 48 but are designed for different pressing loads. In practice, we consider a wide range of values for K_P, K_I and K_{FF} to make it more likely to find good tunings for the system under study. Due to to the fact that the bounds in (8.5) span multiple orders of magnitudes, we optimize $\boldsymbol{x} = \begin{bmatrix} \log_{10} K_P & \log_{10} K_I & \log_{10} K_{FF} \end{bmatrix}^\top$ instead of (8.1).

Black-box optimization. For black-box optimization, we use either $pos\_nmae(\boldsymbol{x})$ in (8.2) or $acc\_std(\boldsymbol{x})$ in (8.4) as the cost function $f(\boldsymbol{x})$ of the GOP (2.1). We do not minimize $err\_max(\boldsymbol{x})$ in (8.3) directly due to the fact that (often) it is in agreement with $pos\_nmae(\boldsymbol{x})$ in (8.2). Roughly speaking, calibrations that achieve good setpoint tracking performances also exhibit low tracking errors (as we will see in Section 8.4). Instead, $err\_max(\boldsymbol{x})$ in (8.3) is better suited for defining a black-box constraint. In particular, in agreement with the industrial standards, we set a threshold of 0.02 meters on the maximum tolerated position tracking error and define Ξ in (2.2) as:

$$\Xi = \{\boldsymbol{x} : err\_max(\boldsymbol{x}) \leq 0.02\}. \tag{8.6}$$

As we will see in Section 8.4, by employing a constrained BBO algorithm with Ξ in (8.6) instead of an unconstrained one ($\Xi = \mathbb{R}^n$), we focus more on those regions of Ω that contain more promising calibrations.

Preference-based optimization. For preference-based optimization, the author of this book plays the role of the calibrator, who expresses the preferences by keeping in mind the control specifications presented in this Section. At each iteration of the PBO procedures, the decision-maker is presented with a *query window*, such as the one in Figure 50, and is asked to express a preference between two calibrations. *We have decided to omit the indicators and the controllers' gains from the query window* (although these can easily be included) *to avoid conditioning the decision-maker into choosing either one of the two calibrations based only on the reported values.* This rationale is also closer to industrial practice where, often, the "goodness" of a tuning depends entirely on the judgement of a calibrator

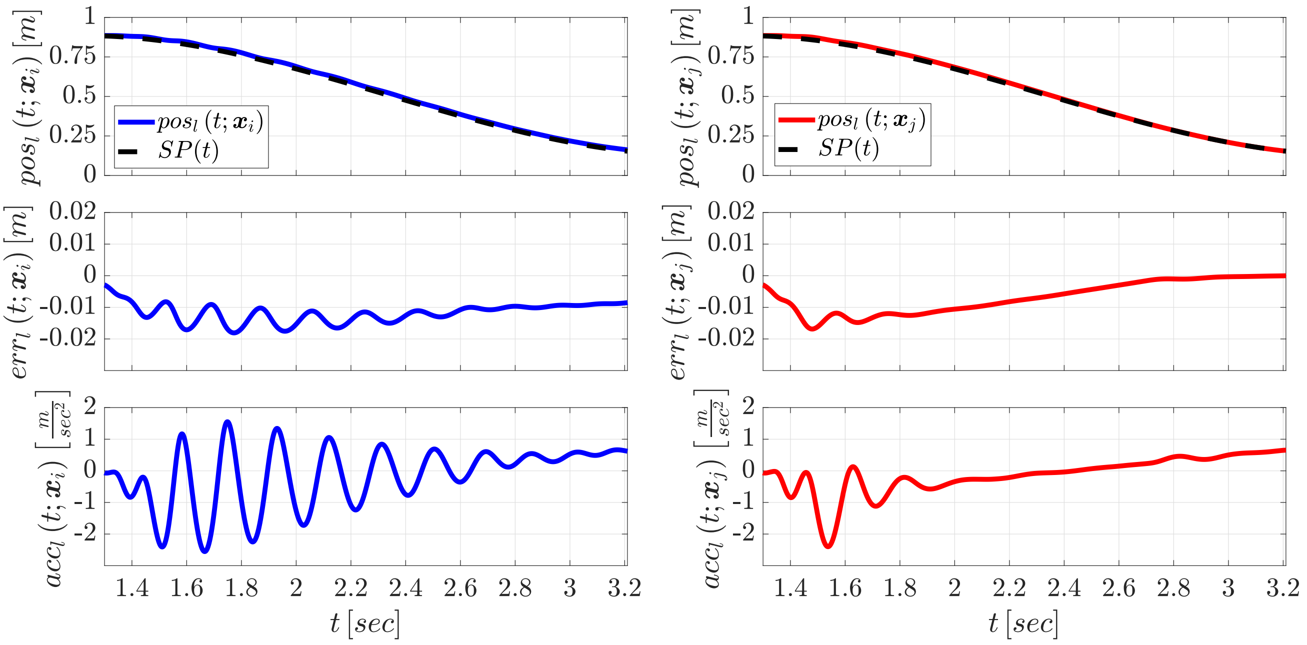


Figure 50: Query window for preference-based optimization. The decision-maker is presented with the position, position tracking error and acceleration signals achieved by two different calibrations, $\boldsymbol{x}_i$ (left) and $\boldsymbol{x}_j$ (right). In this case, tuning $\boldsymbol{x}_j$ is preferred to $\boldsymbol{x}_i$, i.e. $\pi_{\gtrsim}(\boldsymbol{x}_i, \boldsymbol{x}_j) = 1$, since it exhibits fewer acceleration oscillations.

and not on some quantitative indicators. In this case study, we also rely on a decision-maker-based constraint (see Definition 3.6): the calibrator assesses whether a tuning achieves acceptable performances or not. The criteria followed when judging the acceptability of a calibration are: (i) low maximum position tracking error (similarly to (8.6)) and (ii) the acceleration signals must show few to no oscillations. For example, the calibration $\boldsymbol{x}_i$ in Figure 50 is deemed as unacceptable by the decision-maker ($\boldsymbol{x}_i \notin \Xi$), despite it being Ξ-feasible with respect to the black-box constraint in (8.6).

8.4 Experimental results

All the simulations for the hydraulic press under study as well as the black-box and preference-based optimization procedures have been run on the same machine (Intel Core i7 6700HQ @3.50GHz CPU and 32GB of RAM). The simulator of the hydraulic press described in Section 8.2 has been coded in Simulink. Similarly, `GLIS-r` [108], `GLISp-r` [109] (Chapter 5, Algorithm 14), `C-GLIS-r` and `C-GLISp-r` (Chapter 6, Algorithm 16) have been run in MATLAB interpreted code. All the global optimization problems associated to the infill sampling criteria of the procedures have been solved using the `PSWARM` [72] algorithm (see Section 1.2.5). In particular, we have used the MATLAB implementation of `PSWARM` [72] provided by [80, 146, 147][4]. Whenever completely known constraints

[4]Available at `http://www.norg.uminho.pt/aivaz/pswarm/`.

are present (for example, in Problem (6.37)), we have equipped the `PSWARM` [72] procedure with a quadratic penalty function, as described in Section 1.2.6.

8.4.1 Methodology

We take advantage of the simulator available for the hydraulic press under study to test multiple scenarios and get a better glimpse at the performances that the proposed BBO and PBO methods can achieve. We start by tuning PI controllers instead of PI + FF regulators to visualize the results better. We carry out the following optimizations for both PI and PI + FF controllers:

1. Unconstrained ($\Xi = \mathbb{R}^n$) BBO with $f(\boldsymbol{x}) = pos\_nmae(\boldsymbol{x})$ in (8.2) (`GLIS-r` [108]);
2. Unconstrained ($\Xi = \mathbb{R}^n$) BBO with $f(\boldsymbol{x}) = acc\_std(\boldsymbol{x})$ in (8.4) (`GLIS-r` [108]);
3. Unconstrained ($\Xi = \mathbb{R}^n$) PBO with preferences expressed as described in Section 8.3.1 (`GLISp-r` [109]);
4. Constrained BBO with $f(\boldsymbol{x}) = pos\_nmae(\boldsymbol{x})$ in (8.2) and Ξ in (8.6) (`C-GLIS-r`);
5. Constrained BBO with $f(\boldsymbol{x}) = acc\_std(\boldsymbol{x})$ in (8.2) and Ξ in (8.6) (`C-GLIS-r`);
6. Constrained PBO with preferences expressed as described in Section 8.3.1 and Ξ in (8.6) (`C-GLISp-r`);
7. Constrained PBO with preferences expressed as described in Section 8.3.1 and Ξ defined from the decision-maker-based acceptability constraint (`C-GLISp-r`).

A total of 14 optimization problems have been solved. For what concerns preference-based optimization, we point out the following Remark.

Remark 8.1 (Preferences expressed by the decision-maker)**.** *In this case study, since we are dealing with a "real" decision-maker, we have no guarantee that the preferences are always expressed in a consistent fashion. In these experiments, the decisions were taken as methodically as possible, keeping in mind the control specifications described in Section 8.3. In any case, the surrogate model for* $f(\boldsymbol{x})$ *of the GOP* (2.1) *used by `GLISp-r` [109] and `C-GLISp-r` takes into account that human error might be present in the data in* $\mathcal{B}$ (3.9) *(see Section 4.1.2). Thus, it is also interesting to see if the results obtained by the calibrations found through PBO procedures are (somehow) similar. We expect that to be the case if the calibrator has been consistent enough.*

8.4.2 Hyper-parameters for the procedures

The hyper-parameters for the surrogate models and the infill sampling criteria of `GLIS-r` [108], `GLISp-r` [109], `C-GLIS-r` and `C-GLISp-r` have been selected in the same fashion as in Chapter 7 (see the summary in Appendix C). In particular, we have used $\Delta_{cycle} = \langle 0.95, 0.7, 0.35, 0\rangle$ for all the procedures and employed the PSVM classifier (in Section 6.1.3) to estimate the probability of Ξ-feasibility. The only difference from the settings described in Chapter 7 are the iterations at which the shape parameter ϵ_f for $\hat{f}_N\left(\boldsymbol{x}; \boldsymbol{\beta}_f, \epsilon_f\right)$ in (4.1) (PBO) is recalibrated. In particular, we have opted to recalibrate ϵ_f in (4.1) at each iteration of `GLISp-r` [109] and `C-GLISp-r`, i.e.:

$$\mathcal{K}_{R_f} = \{1, 2, \ldots, N_{max} - N_{init}\}\,.$$

The rationale behind this choice is that: (i) we are not solving the same optimization problem multiple times (like we did in Chapter 7, when benchmarking the different algorithms), hence we are not particularly concerned with the computational overhead of the K-fold grid search LOOCV in Algorithm 8; (ii) due to the presence of a "real" decision-maker, we expect the recalibration procedure to help with possible inconsistencies. Instead, similarly to Chapter 7, the shape parameters ϵ_f and ϵ_Ξ for $\hat{f}_N\left(\boldsymbol{x}; \boldsymbol{\beta}_f, \epsilon_f\right)$ in (4.1) (BBO) and $m_{\Xi_N}\left(\boldsymbol{x}; \tilde{\boldsymbol{\beta}}_\Xi, \epsilon_\Xi\right)$ in (6.2), as well as the trade-off parameter C_{SVM} of Problem (6.8), are not recalibrated.

8.4.3 Starting samples, budget and sample evaluations

All the optimization procedures carried out for tuning the parameters of the PI controllers ($n = 2$) are started from $N_{init} = 4$ samples. Instead, for the calibration of the PI + FF controllers ($n = 3$), we set $N_{init} = 8$. Differently from Algorithm 14 and Algorithm 16, we do not use a LHD to generate the initial samples. Instead, we rely on a *full factorial design* (see Section 2.4) since it is closer to industrial practice. Moreover, we use the same starting tunings for all the cases described in Section 8.4.1 to make the comparisons more meaningful.

We have chosen a very limited budget for the optimization procedures employed for the calibration of both the PI and PI + FF controllers: $N_{max} = 50$. That is because, if we were to carry out BBO or PBO on the real hydraulic press instead of its simulator, then 50 sample evaluations would take (roughly) between half a working day and a full working day. Nonetheless, we are also interested in seeing if the proposed algorithms are able to find calibrations that are "good enough" even with such a small budget.

Lastly, for the sake of clarity, we point out that the sample evaluations for the BBO and PBO procedures are carried out as described in Section 8.3.1.

8.4.4 Calibration of the PI controllers

In this Section, we present the results obtained by `GLIS-r` [108], `GLISp-r` [109], `C-GLIS-r` and `C-GLISp-r` when calibrating the parameters of the PI controllers. Figure 51 shows the values of the indicators proposed in Section 8.3 achieved by the best candidates $\boldsymbol{x_{best}}(N)$, $1 \leq N \leq N_{max}$, found by `GLIS-r` [108] and `GLISp-r` [109] (unconstrained optimization) when minimizing $pos\_nmae(\boldsymbol{x})$ in (8.2) and $acc\_std(\boldsymbol{x})$ in (8.4) directly, as well as in the preference-based setting. We can deduce that:

- Unsurprisingly, the curves are monotone decreasing if the corresponding indicator is minimized directly;
- $pos\_nmae(\boldsymbol{x})$ in (8.2) and $err\_max(\boldsymbol{x})$ in (8.3) are mostly in agreement: when minimizing the former, a decrease in the position normalized mean absolute error often leads to a lower maximum position tracking error. That is because one way to achieve tracking performances that are good overall, as described by $pos\_nmae(\boldsymbol{x})$ in (8.2), is to have a small $err_{(\cdot)}(t;\boldsymbol{x}), \forall t = 1, \dots, T$.
- Vice-versa, $pos\_nmae(\boldsymbol{x})$ in (8.2) and $acc\_std(\boldsymbol{x})$ in (8.4) are two (potentially) conflicting criteria. In Figure 51, we can clearly see that a decrease in the latter indicator can lead to a notable increase in the former (similarly for $err\_max(\boldsymbol{x})$ in (8.3)). Situations like this often arise when dealing with control systems for which multiple and often conflicting control specifications are present [84].
- The values of the indicators for $\boldsymbol{x_{best}}(N)$, $1 \leq N \leq N_{max}$, in the preference-based setting do not strictly follow $pos\_nmae(\boldsymbol{x})$ in (8.2) nor $acc\_std(\boldsymbol{x})$ in (8.4). In some sense, the calibrator implicitly makes a trade-off between the two when expressing his preferences.

The position, position tracking error and acceleration signals obtained by the best calibrations, $\boldsymbol{x_{best}}(N_{max})$, found in the unconstrained and constrained BBO and PBO frameworks are depicted in Figure 52. Furthermore, their corresponding values for the indicators are shown in Figure 53. In the PI controller case, except for the calibrations found by minimizing $acc\_std(\boldsymbol{x})$ in (8.4), the tunings obtained by the unconstrained formulations and their respective constrained counterparts are quite similar. In particular:

- The minimization of $pos\_nmae(\boldsymbol{x})$ in (8.2) leads to tunings that show better tracking performances (smaller $err_{(\cdot)}(t;\boldsymbol{x})$) but produce more aggressive accelerations $acc_{(\cdot)}(t;\boldsymbol{x})$;

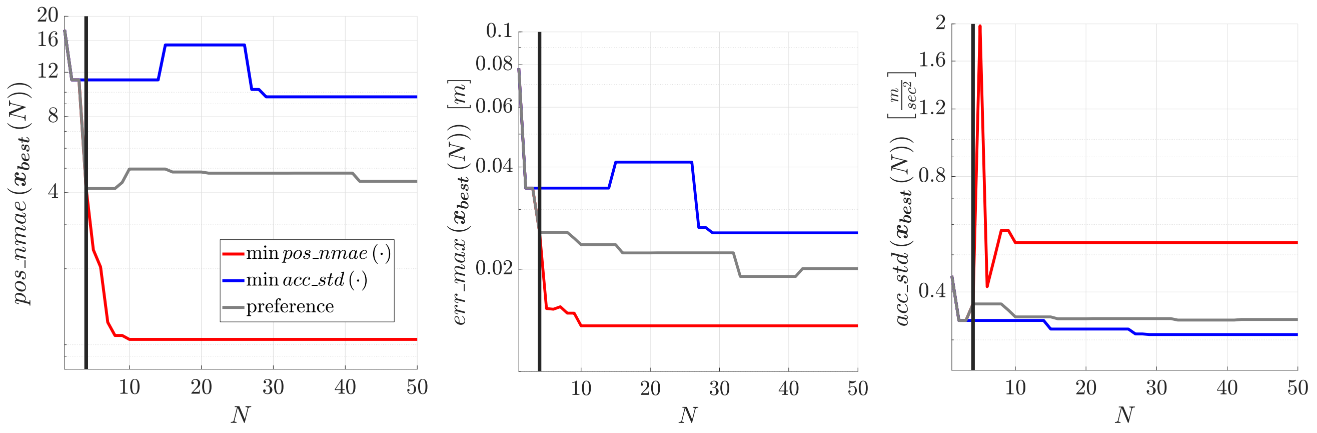


Figure 51: Values of the indicators $pos\_nmae\,(\cdot)$ in (8.2), $err\_max\,(\cdot)$ in (8.3) and $acc\_std\,(\cdot)$ in (8.4) achieved by the best candidate samples $x_{best}\,(N)$, $1 \leq N \leq N_{max}$, found by the unconstrained BBO and PBO procedures when calibrating the PI controllers. Red: indicators for the best candidates found by `GLIS-r` [108] (BBO) when solving $\arg\min_{x\in\Omega} pos\_nmae\,(x)$. Blue: $f\,(x_{best}\,(N))$'s obtained by `GLIS-r` [108] (BBO) when solving $\arg\min_{x\in\Omega} acc\_std\,(x)$. Grey: indicators for the best candidates found by `GLISp-r` [109] (PBO). The black vertical line denotes the number of initial samples N_{init}.

- Vice-versa, the PI calibrations found by solving $\arg\min_{x\in\Omega\cap\Xi} acc\_std\,(x)$ show less pronounced accelerations $acc_{(\cdot)}\,(t;x)$ but higher tracking errors $err_{(\cdot)}\,(t;x)$. In particular, in the unconstrained framework, the resulting tuning shows a maximum position tracking error $err\_max\,(x_{best}\,(N_{max})) > 0.02$ meters. Instead, the calibration found in the constrained setting satisfies the black-box constraint in (8.6) at the cost of a higher $acc\_std\,(x_{best}\,(N_{max}))$ (0.310, unconstrained, vs 0.331 meters per seconds squared, constrained).

- Lastly, the calibrations found by means of preference-based optimization achieve a good balance between position setpoint tracking and moderate acceleration signals. To confirm this, look at the values of the indicators for $x_{best}\,(N_{max})$ resulting from PBO in Figure 53, which are in-between those obtained from minimizing $pos\_nmae\,(x)$ in (8.2) and $acc\_std\,(x)$ in (8.4) directly.

So far, it seems like adding the black-box constraint in (8.6) does not change the outcomes of the optimization procedures by much. However, there is a "hidden" advantage to the constrained BBO or PBO problems over the unconstrained ones: *we test fewer calibrations that exhibit unsatisfactory performances*. That is because we limit the exploration of the whole feasible region Ω of the GOP (2.1), focusing more on those zones that are likely to contain Ξ-feasible samples. In Figure 54, we show the box plots related to the maximum position tracking errors achieved by all the tunings tested by the BBO and PBO procedures. In the unconstrained framework, a good portion of the calibrations lead to a $err\_max\,(x_i) > 0.02$ meters. Many even exceed 0.1 meters, which is clearly unacceptable since the setpoint $SP\,(t)$ ranges between 0 and 1 meter (as in Figure 52). Instead, very few calibrations tested by the constrained BBO and PBO procedures actually violate the black-box constraint in (8.6).

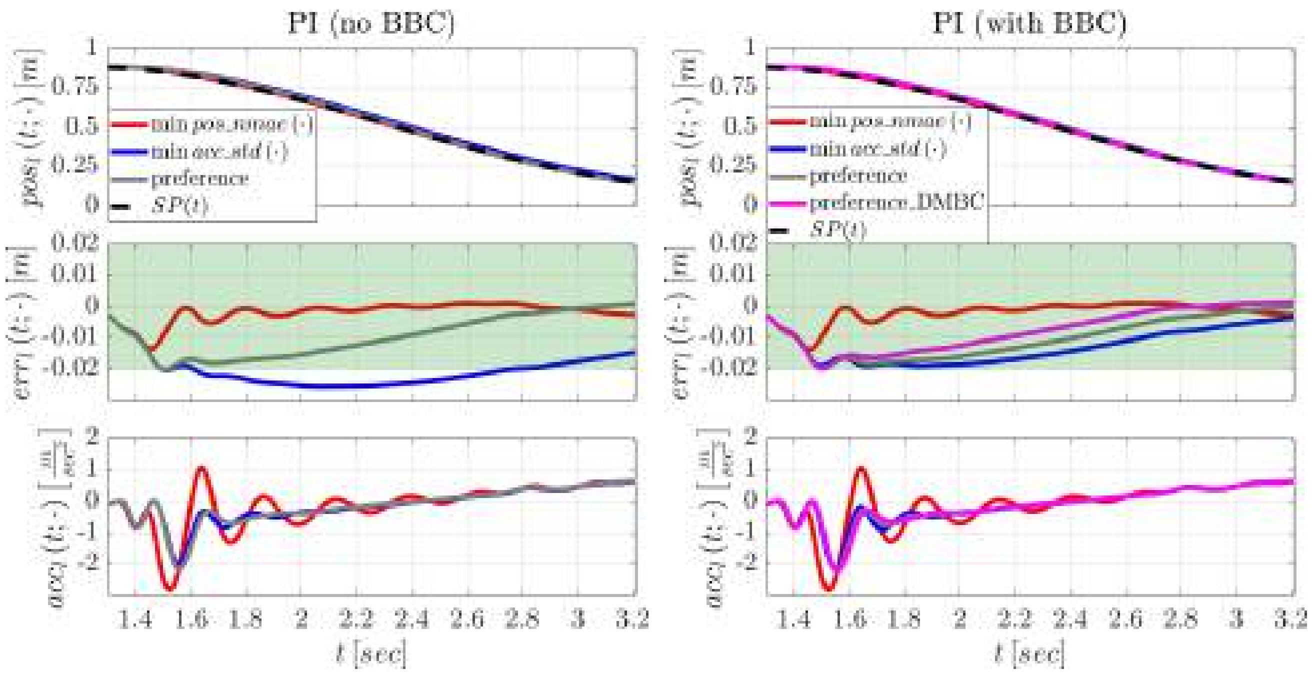


Figure 52: Position $pos_l(t;\cdot)$, position error $err_l(t;\cdot)$ and acceleration $acc_l(t;\cdot)$ signals obtained by the best calibrations $x_{best}(N_{max})$ found by the unconstrained (left) and constrained (right) black-box and preference-based optimization procedures when calibrating the PI controllers. The shaded green area corresponds to $|err_l(t,x)| \leq 0.02$ meters and is related to the black-box constraint in (8.6). Red: signals resulting from the calibrations found by GLIS-r [108] or C-GLIS-r (BBO) when solving $\arg\min_{x\in\Omega\cap\Xi} pos\_nmae(x)$. Blue: performances of $x_{best}(N_{max})$ obtained by GLIS-r [108] or C-GLIS-r (BBO) when solving $\arg\min_{x\in\Omega\cap\Xi} acc\_std(x)$. Grey: signals associated to the calibrations found by GLISp-r [109] or C-GLISp-r (PBO); in the constrained case, we consider Ξ defined as in (8.6). Magenta: signals resulting from $x_{best}(N_{max})$ obtained through C-GLISp-r when a decision-maker-based constraint is present.

Figure 60 shows the surrogate models for the cost functions $f(x)$ and the black-box constraints functions of the GOPs (2.1) in the different settings described in Section 8.4.1. We can clearly see that, in the unconstrained black-box optimization framework, GLIS-r [108] spends a lot of the search effort on exploring the whole set Ω. The same can be said for GLISp-r [109] (unconstrained PBO), although this behavior is much less pronounced. That is because GLISp-r [109] often returns new candidate samples that improve upon the current best one when δ in (5.12) is high (exploitation). Hence, due to the greedy δ-cycling strategy (Section 5.2.3), the procedure spends more time exploiting than exploring. Instead, in the constrained framework, exploration is limited and most tunings tend to have proportional gains K_P that are close to its corresponding upper bound in (8.5). That is because, in order to satisfy the black-box constraint $err\_max(x) \leq 0.02$ meters, we must lean towards more aggressive controllers. However, *the constraint on the maximum position tracking error does not take into account that high K_P's and K_I's can lead to pronounced acceleration oscillations (which must be avoided)*, see for example Figure 50 - tuning x_i. Yet, it is much harder to define a suitable constraint for the latter objective. It is easier to let a decision-maker express whether a tuning is acceptable or

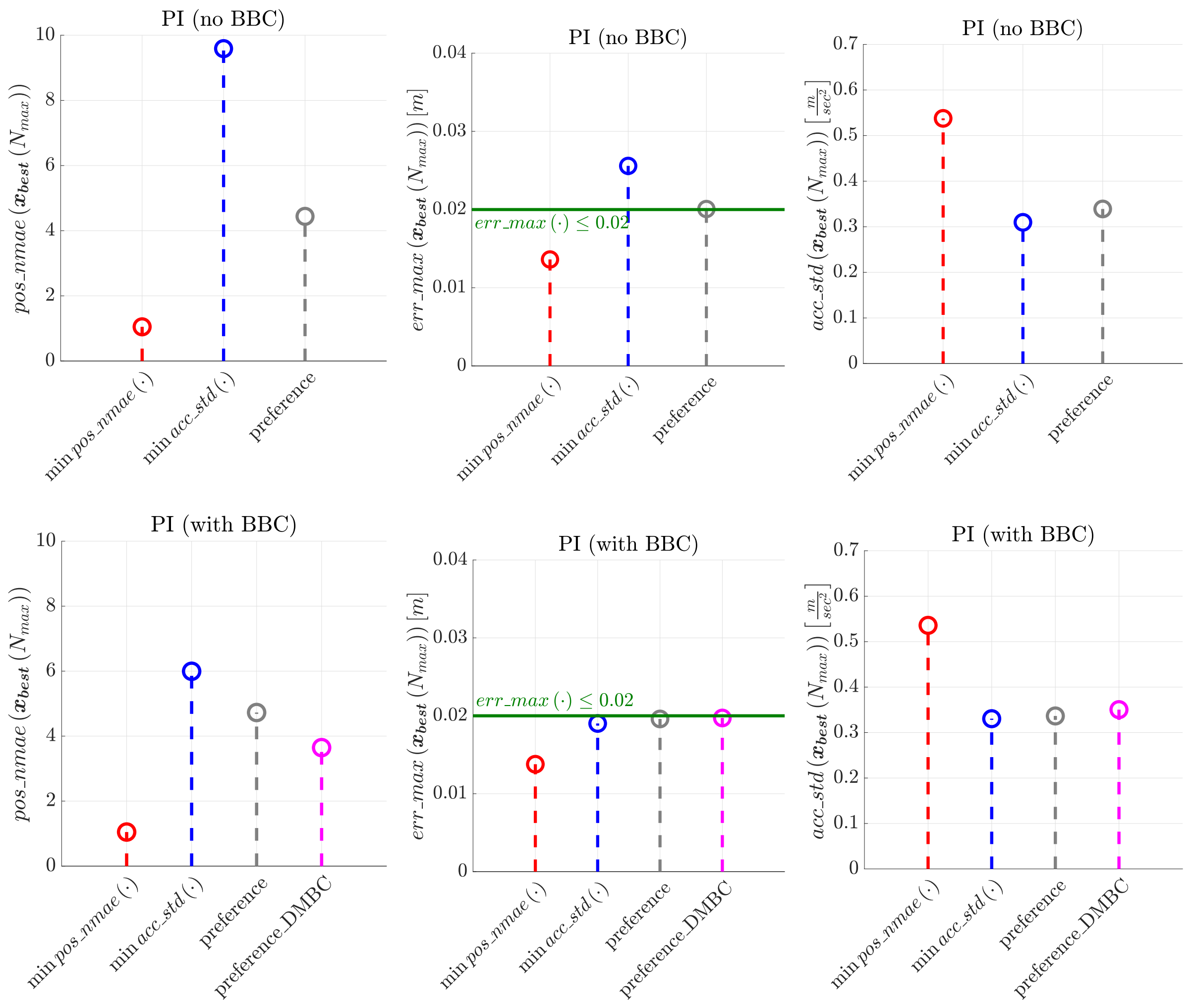


Figure 53: Values of the performance indicators achieved by the best calibrations $x_{best}\left(N_{max}\right)$ of the PI controllers found by the different strategies (unconstrained on the top and constrained on the bottom). The colors follow the same scheme described in Figure 52.

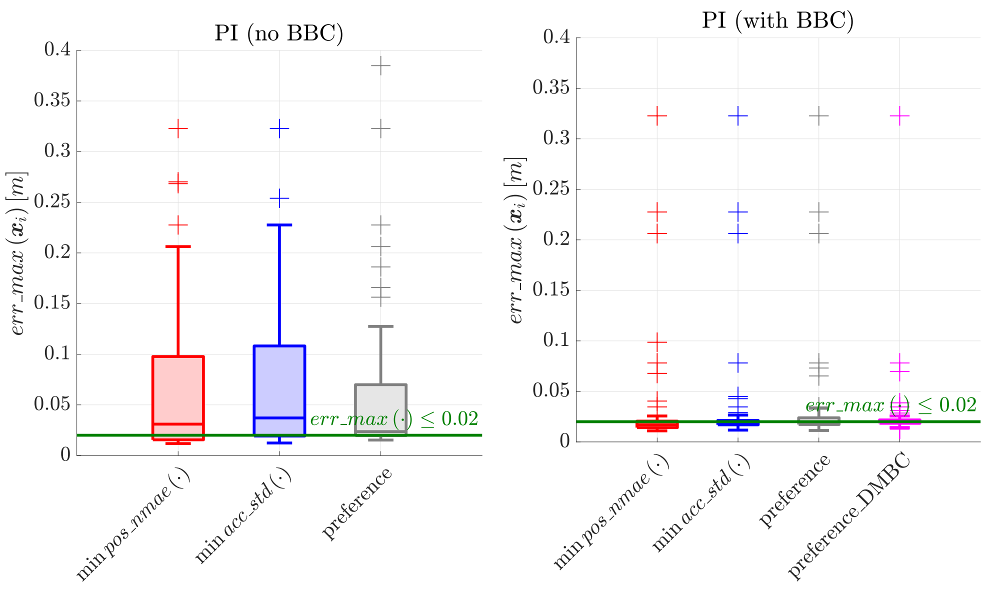


Figure 54: Box plots that describe the maximum position tracking errors achieved by all the PI controllers' calibrations tried by the unconstrained (left) and constrained (right) BBO and PBO procedures. The colors follow the same scheme described in Figure 52.

not, as we did in the context of PBO with the addition of the decision-maker-based constraint. As a matter of fact, the green crosses in Figure 60 depict those calibrations that lead to unacceptable acceleration oscillations, which are deemed as Ξ-infeasible by the DM; notice how all of them have high K_P's. Consequently, the decision boundaries of the PSVM classifiers for Ξ defined as in (8.6) and when, instead, the Ξ-feasibility is determined by the calibrator, are different. As a final remark, we point out that the surrogate models for the scoring function $f(\boldsymbol{x})$ of the decision-maker in Figure 60 differ quite a lot in the three cases. That is due to the fact that, in the unconstrained framework, at the last iteration of `GLISp-r` [109] the recalibration procedure selects $\epsilon_f = 10$, which results in a "more local" $\hat{f}_N(\boldsymbol{x})$ in (4.1). Instead, the opposite can be said for the surrogates in the constrained framework, where we have $\epsilon_f = 0.1$. In any case, most samples tried by the PBO procedures are such that $\log_{10} K_P \geq -0.1$ (i.e. $K_P \geq 0.8$) and the best candidates are similar. Hence, we can conclude that the calibrator has been sufficiently consistent with his choices (see Remark 8.1).

8.4.5 Calibration of the PI + FF controllers

Now, we address the performances achieved by the calibrations of the PI + FF controllers found by `GLIS-r` [108], `GLISp-r` [109], `C-GLIS-r` and `C-GLISp-r`. We also report a summary table (Table 8) that shows the best tunings obtained by the strategies proposed in Section 8.4.1, their corresponding indicators' values and the execution times required for performing BBO and PBO with a budget of $N_{max} = 50$ sample evaluations.

We start by analyzing the signals associated to $\boldsymbol{x_{best}}(N_{max})$ returned by the several optimization procedures, which are depicted in Figure 55. Overall, compared to the results achieved by the PI controllers (in Figure 52), the PI + FF regulators attain better setpoint tracking performances (lower $err_{(\cdot)}(t;\boldsymbol{x_{best}}(N_{max}))$). That is to be expected since the feed forward action makes the regulators more responsive and able to react quicker to setpoint changes. However, as a consequence, the control actions tend to be less moderate, which could result in oscillatory accelerations. As a matter of fact, in the unconstrained framework, the best calibrations found when minimizing $pos\_nmae(\boldsymbol{x})$ in (8.2) and $acc\_std(\boldsymbol{x})$ in (8.4) directly (BBO) obtain similar $err_{(\cdot)}(t;\boldsymbol{x_{best}}(N_{max}))$ and $acc_{(\cdot)}(t;\boldsymbol{x_{best}}(N_{max}))$ signals. In particular, both acceleration signals show comparable oscillations. Surprisingly, in Table 8, we can see that the PI + FF controller achieves a lower $acc\_std(\boldsymbol{x_{best}}(N_{max}))$ than its PI counterpart when the latter indicator is minimized directly in the unconstrained framework (0.310, PI, vs 0.288 meters per seconds squared, PI + FF), despite its corresponding $acc_{(\cdot)}(t;\boldsymbol{x_{best}}(N_{max}))$ signals show more oscillations. In practice, the indicator $acc\_std(\boldsymbol{x})$ in (8.4) might not be completely suited to describe the third control specification presented in Section 8.3. That is because even an oscillatory signal can have a low standard deviation if the amplitude of the oscillations is small enough. *This is one of the main difficulties encountered by BBO procedures when applied to control systems applications: translating qualitative control specifications into suitable performance indicators is (often) not straightforward. PBO methods are not affected by this shortcoming since they rely only on a calibrator who typically has a clear objective in mind.* As a matter of fact, the performances of the calibrations found through preference-based optimization, reported in Figure 55, better capture the control specifications described in Section 8.3.

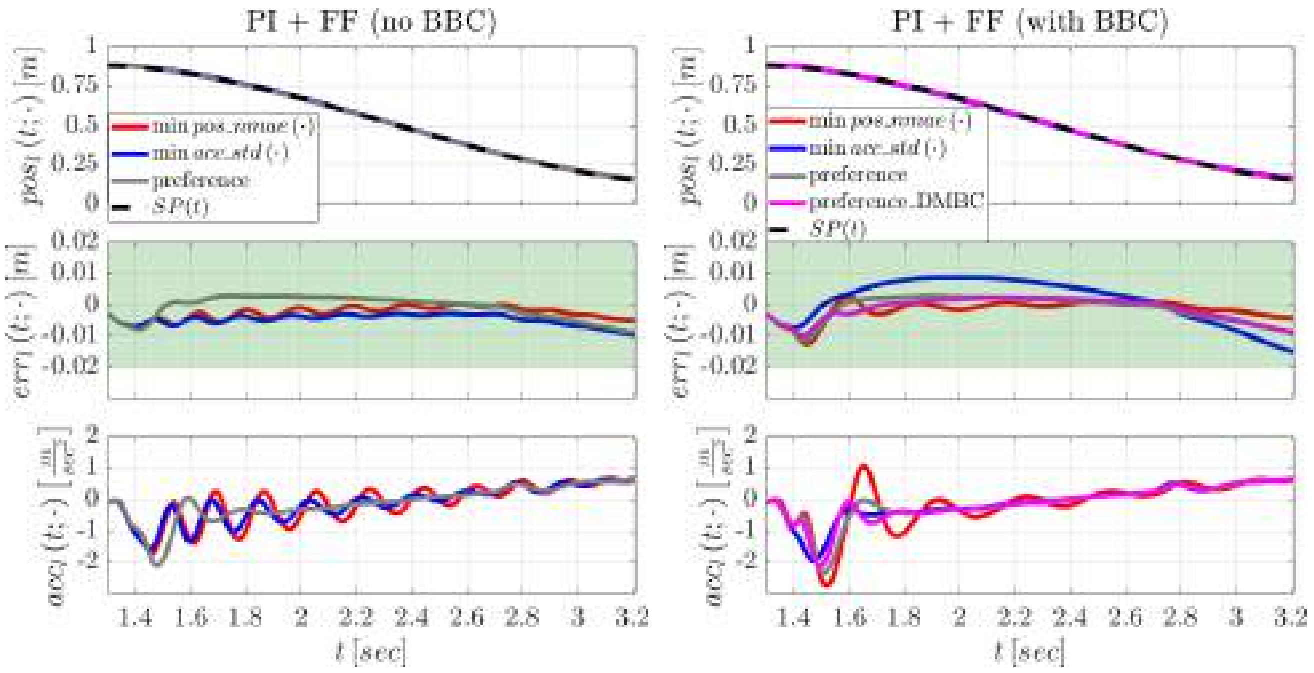


Figure 55: Position $pos_l(t;\cdot)$, position error $err_l(t;\cdot)$ and acceleration $acc_l(t;\cdot)$ signals obtained by the best calibrations $\boldsymbol{x_{best}}(N_{max})$ found by the unconstrained (left) and constrained (right) black-box and preference-based optimization procedures when calibrating the PI + FF controllers. The shaded green area corresponds to $|err_l(t,\boldsymbol{x})| \leq 0.02$ meters and is related to the black-box constraint in (8.6). The colors follow the same scheme described in Figure 52.

If we take a closer look at the best calibrations found when minimizing $pos\_nmae(\boldsymbol{x})$ in (8.2) directly, in the unconstrained BBO framework, we notice in Table 8 that $pos\_nmae(\boldsymbol{x_{best}}(N_{max}))$ for the PI + FF controller is higher than its PI regulator counterpart (1.241, PI + FF, vs 1.050, PI). The addition of the feed forward action should improve the setpoint tracking performances; hence, we would expect the opposite result. However, the issue here is that the budget $N_{max} = 50$ is quite restrictive for the black-box optimization of $pos\_nmae(\boldsymbol{x})$ in (8.2). As a matter of fact, we have tried increasing the budget to $N_{max} = 200$ sample evaluations and ran `GLIS-r` [108] once again, both for calibrating the PI and the PI + FF controllers. The resulting convergence plots are depicted in Figure 56. In the PI regulator case, `GLIS-r` [108] is able to find a tuning that minimizes $pos\_nmae(\boldsymbol{x})$ in (8.2) quite fast, in about 10 sample evaluations; the sample evaluations for $10 < N \leq 200$ carry no improvement. Instead, when tuning the PI + FF controllers, the optimization proves to be much harder (partially because we have to explore a three-dimensional space instead of a two-dimensional one). After roughly 80 sample evaluations, the performances of the best PI + FF controller calibration are better than those of its PI counterpart. Moreover, GLIS-r [108] keeps improving upon $pos\_nmae\,(\boldsymbol{x})$ in (8.2) as the iterations go on, achieving, at last, $pos\_nmae\,(\boldsymbol{x_{best}}\,(N_{max})) = 0.934$ against $pos\_nmae\,(\boldsymbol{x_{best}}\,(N_{max})) = 1.050$ of the PI controller.

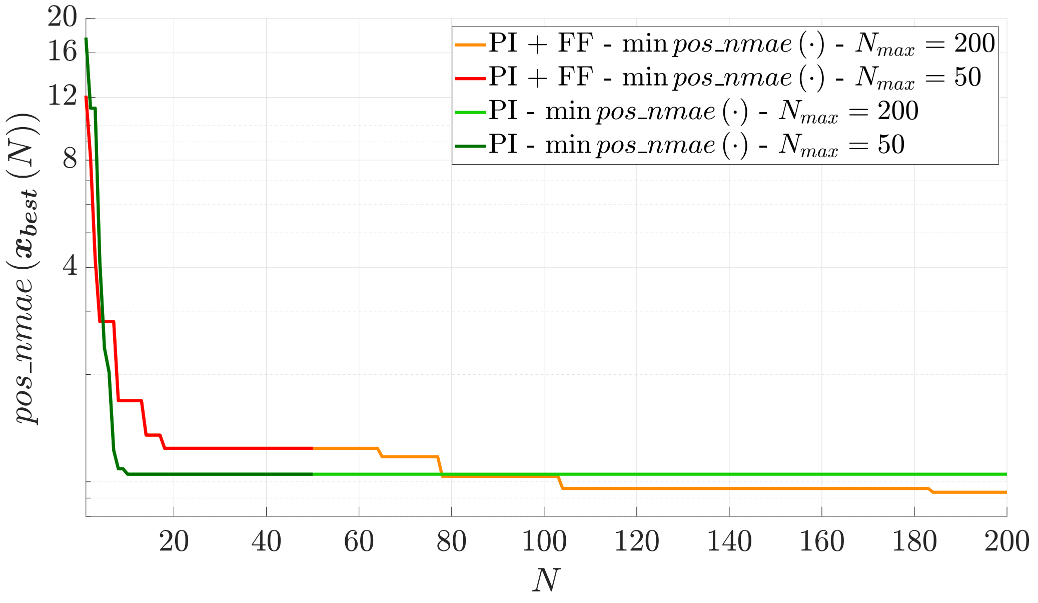


Figure 56: Convergence plots of GLIS-r [108] when minimizing the cost function $pos\_nmae\,(\boldsymbol{x})$ in (8.2) directly, in the unconstrained BBO framework. We compare the performances achieved by the PI and the PI + FF controllers when the budgets are $N_{max} = 50$ and $N_{max} = 200$ respectively.

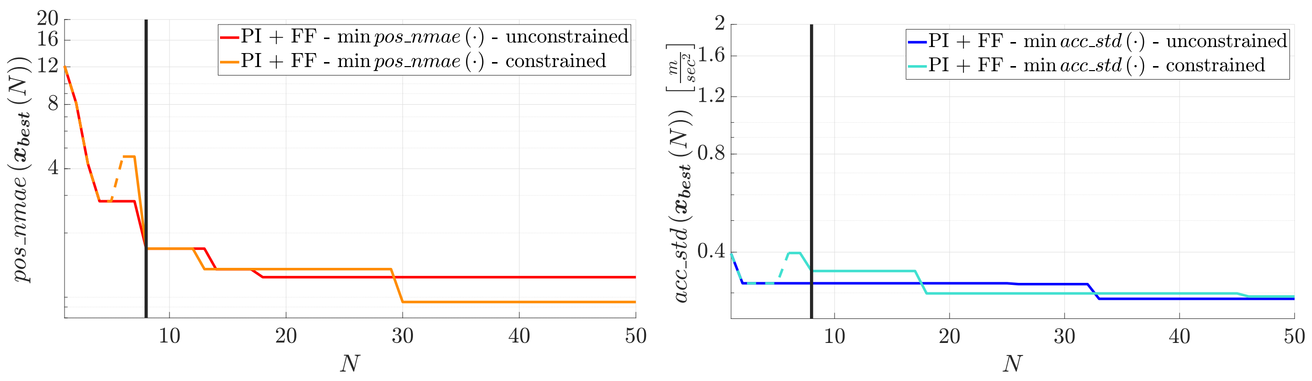


Figure 57: Convergence plots of GLIS-r [108] and C-GLIS-r when minimizing the cost functions $pos\_nmae\,(\boldsymbol{x})$ in (8.2) (left) and $acc\_std\,(\boldsymbol{x})$ in (8.4) (right) directly, both in the unconstrained and constrained BBO frameworks. For the latter case, Ξ is defined as in (8.6). The black vertical line denotes the number of initial samples N_{init}. As long as C-GLIS-r has not found a Ξ-feasible sample, the corresponding curves $f\,(\boldsymbol{x_{best}}\,(N))$ are depicted with dashed lines; we switch to continuous lines when $\boldsymbol{x_{best}}\,(N) \in \Xi$. Note that, despite GLIS-r [108] and C-GLIS-r start from the same samples, $f\,(\boldsymbol{x_{best}}\,(N))$ differ for $N \leq N_{init}$. That is because, in the constrained framework, a Ξ-feasible sample always improves upon a Ξ-infeasible one.

Figure 58 depicts the indicators' values for the best calibrations of the PI + FF regulators found by the different strategies described in Section 8.4.1. Overall, compared to the results obtained by the PI controllers in Figure 53, the setpoint tracking performances achieved by the PI + FF controllers are better, as we would expect due to the addition of the feed forward action. As a matter of fact, even when $pos\_nmae\left(\boldsymbol{x}\right)$ in (8.2) is not minimized directly, the best calibrations found for the PI + FF regulators mostly exhibit lower $pos\_nmae\left(\boldsymbol{x_{best}}\left(N_{max}\right)\right)$ than those in Figure 53. Furthermore, in any case, the resulting $err\_max\left(\boldsymbol{x_{best}}\left(N_{max}\right)\right)$ is lower for the PI + FF controllers, regardless of whether the black-box constraint $err\_max\left(\boldsymbol{x}\right) < 0.02$ meters is present or not. Lastly, for what concerns $acc\_std\left(\boldsymbol{x}\right)$ in (8.4), both regulators achieve comparable values.
When calibrating the PI + FF controllers, the addition of the black-box constraint on the maximum position tracking error, in (8.6), brings two improvements:

1. Similarly to what we have seen in Section 8.4.4, fewer unsatisfactory calibrations are tried, as depicted by the box plots of $err\_max\left(\boldsymbol{x_i}\right)$ in Figure 59;

2. *By reducing the search effort in those regions of Ω that are likely to contain Ξ-infeasible samples, the constrained BBO and PBO procedures can achieve similar (or even better) results to those of their unconstrained counterparts in fewer sample evaluations.* Figure 57 shows the convergence plots of `GLIS-r` [108] and `C-GLIS-r` when minimizing $pos\_nmae\left(\boldsymbol{x}\right)$ in (8.2) and $acc\_std\left(\boldsymbol{x}\right)$ in (8.4) directly. For the former indicator, `C-GLIS-r` is even able to reach a value of $pos\_nmae\left(\boldsymbol{x_{best}}\left(N_{max}\right)\right)$ that is comparable to that achieved by `GLIS-r` [108] when $N_{max} = 200$ (cf. Figure 56). Instead, for the latter indicator, the best candidate found by `C-GLIS-r` is slightly worse than that of `GLIS-r` [108]. That is because the black-box constraint $err\_max\left(\boldsymbol{x}\right) < 0.02$ meters favors better setpoint tracking performances (small $pos\_nmae\left(\boldsymbol{x}\right)$) but does not necessarily drive the search towards those regions of Ω where $acc\_std\left(\boldsymbol{x}\right)$ is low.

We conclude this Section by taking a look at the execution times of the optimization procedures (reported in Table 8), which range between 18 and 40 minutes. These exceed the execution times observed when solving the benchmark optimization problems in Chapter 7 by quite a lot (cf. Table 4 and Table 7). In fact, the computational overhead of the BBO and PBO procedures is practically negligible when compared to time required to perform the simulations on the hydraulic press under study. Moreover, if we were to perform the same experiments on the real system, the optimizations would take far more than 40 minutes. This result is inline with the main assumption of surrogate-based methods (Assumption 2.3), which says that sample evaluations are the most time-consuming operations performed during the optimization procedure. As a final remark, note that the execution

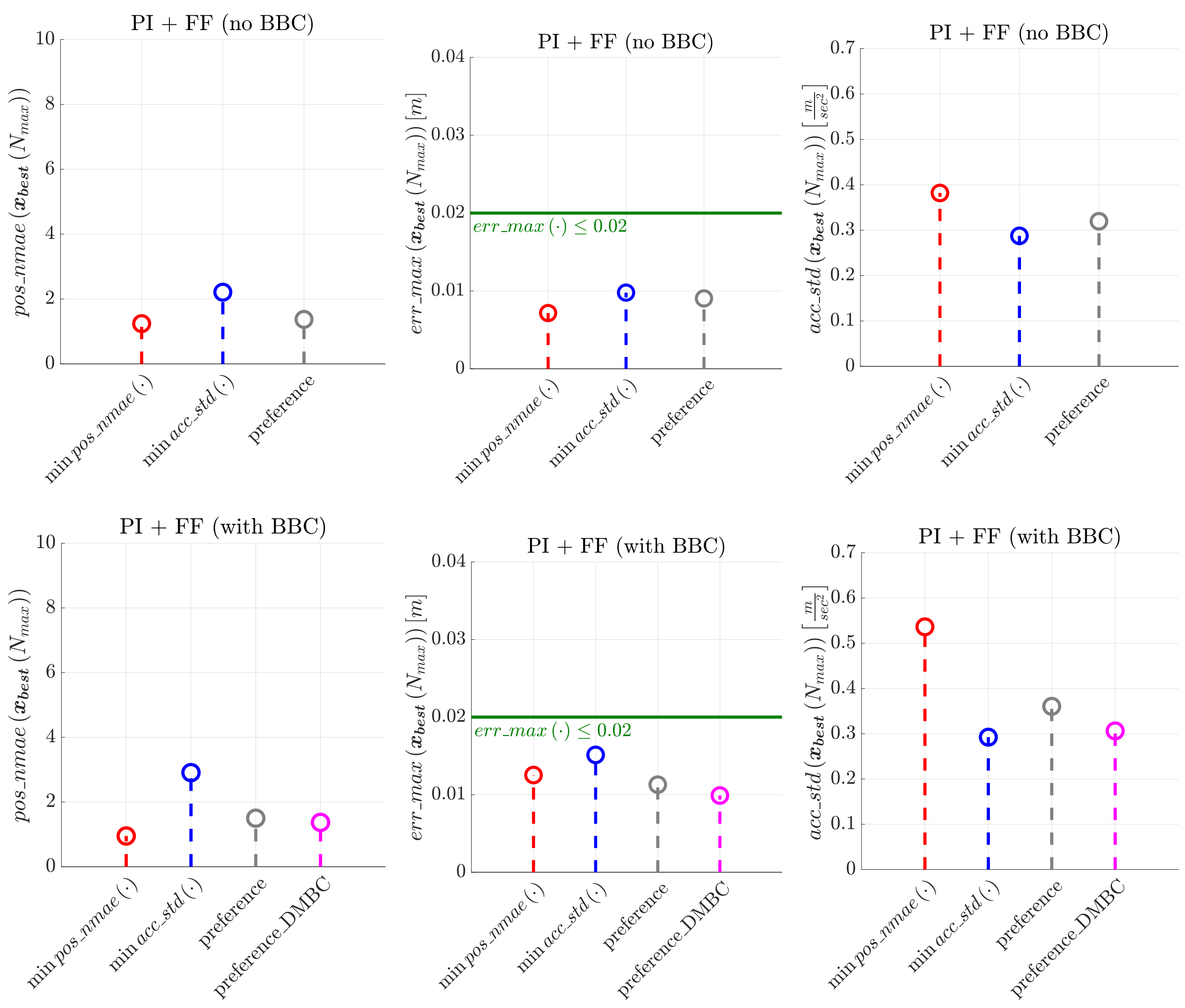


Figure 58: Values of the performance indicators achieved by the best calibrations $x_{best}\left(N_{max}\right)$ of the PI + FF controllers found by the different strategies (unconstrained on the top and constrained on the bottom). The colors follow the same scheme described in Figure 52.

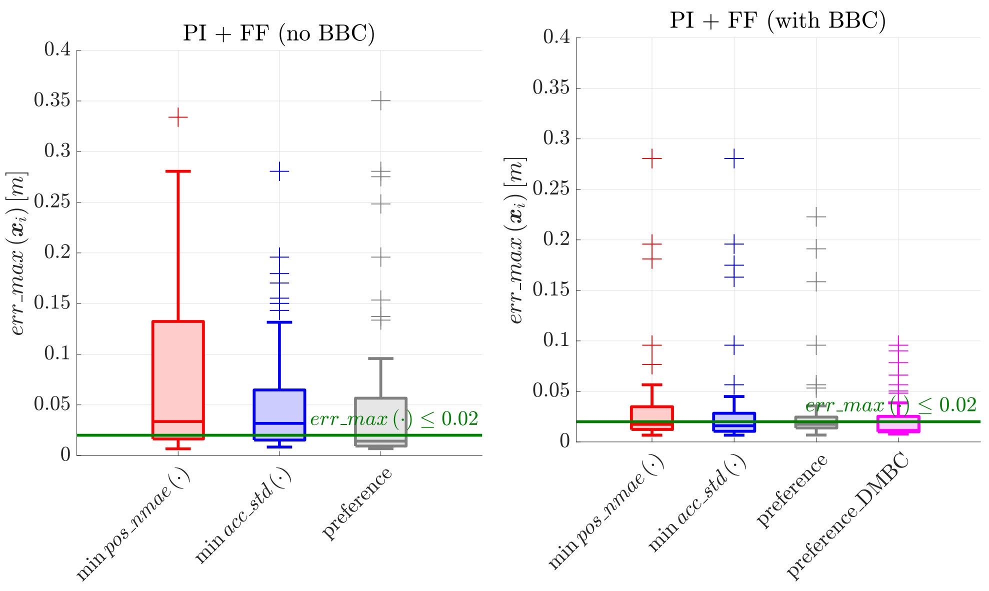
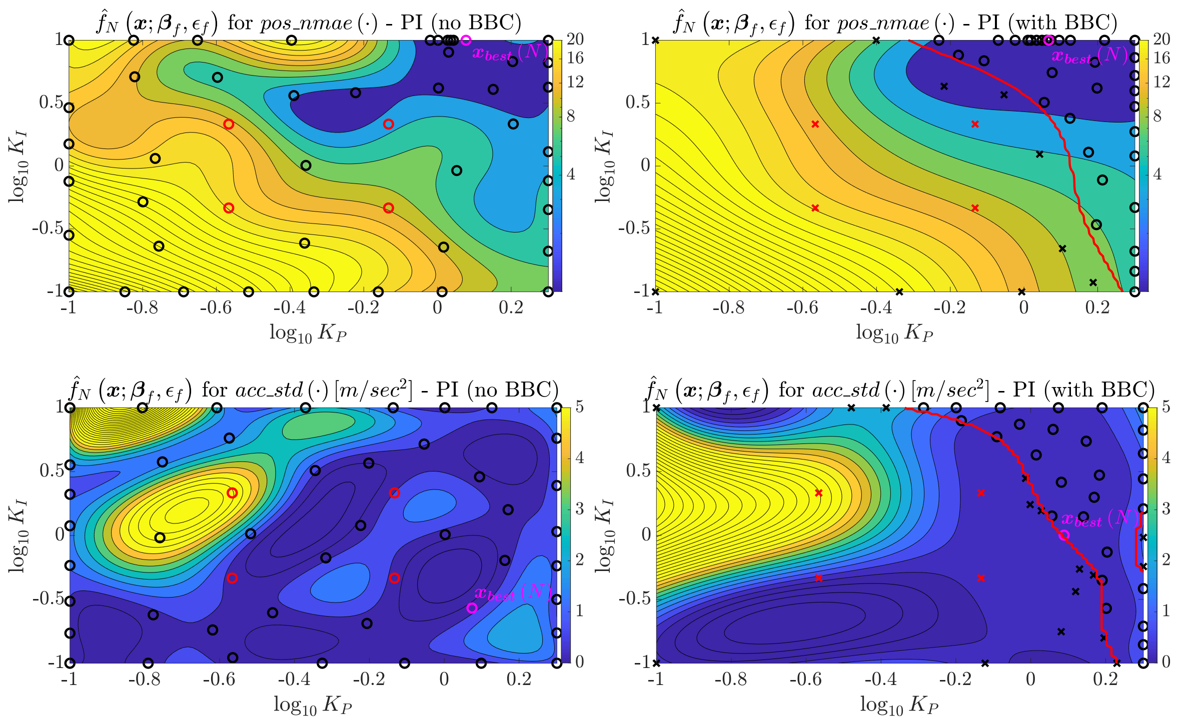
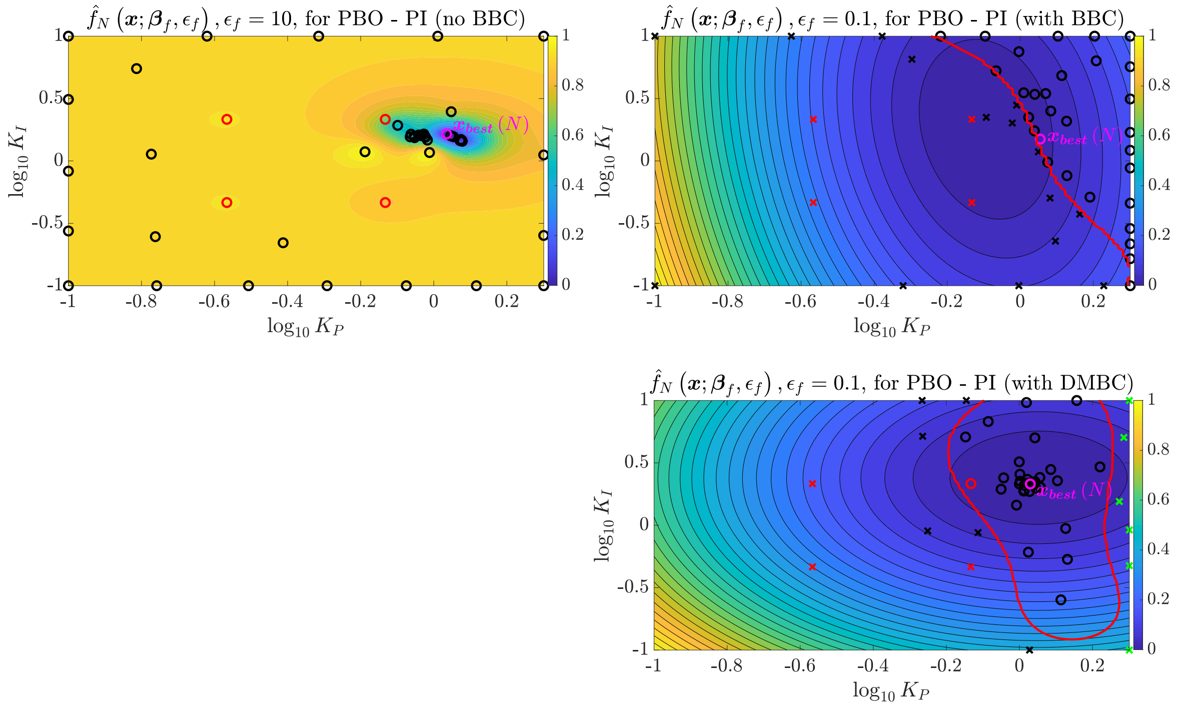


Figure 59: Box plots that describe the maximum position tracking errors achieved by all the PI + FF controllers' calibrations tried by the unconstrained (left) and constrained (right) BBO and PBO procedures. The colors follow the same scheme described in Figure 52.

times of the PBO algorithms exceed those of their BBO counterparts. That is because they must wait for the responses of the decision-maker, who interacts with the procedures through the query window in Figure 50.



(a) Unconstrained (left) and constrained (right) BBO with $f(\boldsymbol{x}) = pos\_nmae(\boldsymbol{x})$ in (8.2) (top) and $f(\boldsymbol{x}) = acc\_std(\boldsymbol{x})$ in (8.4) (bottom).



(b) Unconstrained (left) and constrained (right) preference-based optimization. The bottom row shows the results of PBO when equipped with the decision-maker-based constraint. The three surrogates are made comparable by rescaling them to the $[0,1]$ range.

Figure 60: Comparison of the surrogate models for the cost functions $f(x)$ of the GOPs (2.1) used to to calibrate the PI controllers. The surrogate models $\hat{f}_N(x)$ in (4.1) are those at the last iterations of the `GLIS-r` [108], `GLISp-r` [109], `C-GLIS-r` and `C-GLISp-r` procedures ($N = N_{max} - 1 = 49$). The red curves constitute the decision boundaries of the PSVM classifiers (Section 6.1.3). The tried calibrations are depicted as circles (Ξ-feasible) or crosses (Ξ-infeasible). The red points are the N_{init} initial samples. Finally, the best candidates are highlighted in magenta.

Table 8: Summary table that shows the calibrations of the PI and PI + FF controllers obtained by the different strategies proposed in Section 8.4.1. For each tuning, we report its corresponding values for the indicators $pos\_nmae\,(\boldsymbol{x})$ in (8.2), $err\_max\,(\boldsymbol{x})$ in (8.3) and $acc\_std\,(\boldsymbol{x})$ in (8.4), highlighting in bold font which calibrations achieved the best results. We also report the execution time required to find each tuning.

Optimization problem			**Best calibration**			**Indicator at $\boldsymbol{x_{best}}\,(N_{max})\,, N_{max} = 50$**			
$f\,(\boldsymbol{x})$	Ξ	**Controller**	K_P	K_I	K_{FF}	$pos\_nmae\,(\cdot)$	$err\_max\,(\cdot)\;[m]$	$acc\_std\,(\cdot)\;\boxed{\frac{m}{sec^2}}$	**Execution times** $[sec]$
$pos\_nmae\,(\boldsymbol{x})$	$\mathbb{R}^n$	PI	1.195	10.000	/	1.050	0.014	0.538	1454 (24.2 *min*)
		PI + FF	2.000	3.479	0.041	1.241	**0.007**	0.382	1127 (18.8 *min*)
	in (8.6)	PI	1.168	10.000	/	1.048	0.014	0.536	1410 (23.5 *min*)
		PI + FF	0.968	10.000	0.024	**0.950**	0.013	0.536	1377 (23.0 *min*)
$acc\_std\,(\boldsymbol{x})$	$\mathbb{R}^n$	PI	1.186	0.270	/	9.588	0.026	0.310	1164 (19.4 *min*)
		PI + FF	2.000	0.100	0.055	2.210	0.010	**0.288**	1350 (22.5 *min*)
	in (8.6)	PI	1.228	1.006	/	5.996	0.019	0.331	1188 (19.8 *min*)
		PI + FF	0.939	2.223	0.1	2.909	0.015	0.293	1360 (22.7 *min*)
preference	$\mathbb{R}^n$	PI	1.084	1.645	/	4.441	0.020	0.339	2204 (36.7 *min*)
		PI + FF	1.072	5.084	0.077	1.372	0.009	0.320	2437 (40.6 *min*)
	in (8.6)	PI	1.134	1.496	/	4.726	0.019	0.337	2061 (34.4 *min*)
		PI + FF	0.807	3.979	0.069	1.499	0.011	0.361	2075 (34.6 *min*)
	DMB	PI	1.071	2.141	/	3.646	0.020	0.350	2154 (35.9 *min*)
		PI + FF	1.103	2.618	0.068	1.371	0.010	0.306	2331 (38.9 *min*)

8.5 Concluding remarks

This case study has taught us many valuable lessons on black-box and preference-based optimization procedures when applied to control systems applications. First and foremost, PBO can be particularly useful when only qualitative control specifications are available, saving us the time of designing suitable performance indicators. Similarly, decision-maker-based constraints prove to be quite handy when dealing with requirements that are hard to define analytically.
Concerning the performances of the calibrations found by the strategies proposed in Section 8.4.1, most of the time the PI + FF controllers outperform the PI regulators, when tuned properly. This is not surprising since the former controllers are more complex and can react to setpoint changes much quicker than the latter. However, tuning $n = 3$ parameters instead of just $n = 2$ might require more experiments, especially in the unconstrained BBO and PBO frameworks. As we have seen in Chapter 1, finding the global solutions of an optimization problem becomes much harder as the dimensionality of the decision vector increases. Hence, when the budget N_{max} is quite restrictive, it might be a good idea to include black-box constraints to limit the search of Ω or, alternatively, to spend more time exploiting than exploring.
Lastly, we have seen that, even though surrogate-based methods are often computationally expensive, the computational overhead added by the optimization procedures is practically negligible compared to the time required to perform sample evaluations that involve computer simulations.

8.5.1 A note on preference-based and multi-objective optimization

As in many control systems applications, some of the control specifications presented in Section 8.3 are conflicting. In our case, there does not exist a calibration that simultaneously minimizes both $pos\_nmae\left(\boldsymbol{x}\right)$ in (8.2) and $acc\_std\left(\boldsymbol{x}\right)$ in (8.4). Instead of minimizing the two indicators separately, we could define the following *Multi-Objective Optimization (MOO) problem* [84]:

$$\begin{aligned} \mathcal{X}^* = \arg\min_{\boldsymbol{x}} & \begin{bmatrix} pos\_nmae\left(\boldsymbol{x}\right) \\ acc\_std\left(\boldsymbol{x}\right) \end{bmatrix} \\ \text{s.t.} \quad & \boldsymbol{x} \in \Omega \cap \Xi. \end{aligned} \tag{8.7}$$

In order to solve Problem (8.7), a black-box multi-objective optimization algorithm must be used, such as the one proposed in [113]. Similarly to preference-based optimization, human decision-makers also play a key role in MOO:

- *A-priori MOO* methods require that the decision-maker expresses sufficient "preference information" on the different objectives. For example, his/her "preferences" can be specified in the form of weights, $\omega_{nmae} \in \mathbb{R}_{>0}$ and $\omega_{std} \in \mathbb{R}_{>0}$, associated to $pos\_nmae\left(\boldsymbol{x}\right)$ in (8.2) and $acc\_std\left(\boldsymbol{x}\right)$ in (8.4) respectively. Problem (8.7) is then scalarized into a single objective optimization problem with cost function:

$$f\left(\boldsymbol{x}\right) = \omega_{nmae} \cdot pos\_nmae\left(\boldsymbol{x}\right) + \omega_{std} \cdot acc\_std\left(\boldsymbol{x}\right). \tag{8.8}$$

There also exist several other ways of scalarizing a multi-objective optimization problem, see [84].

- *A-posteriori MOO* methods, such as the one in [113], return a set of Pareto-optimal solutions for Problem (8.7). The decision-maker is then asked to choose a calibration among the ones supplied by the optimization procedure.

The main advantages of preference-based optimization over black-box multi-objective optimization are: (i) *there is no need to explicitly define multiple cost functions* since PBO methods estimate $f\left(\boldsymbol{x}\right)$ directly from the preferences expressed by the DM; (ii) *the conflicting criteria are implicitly taken into account by the decision-maker* without the need, for example, to choose the weights in (8.8).

Conclusions

This book dealt with the problem of finding the global solution(s) of an optimization problem whose cost function and (possibly but not necessarily) some of the constraints functions are unknown and expensive to evaluate. The most suited algorithms for such task are surrogate-based methods, which rely on approximations of the latent functions to drive the search towards the minimizer(s) of the optimization problem. We have considered both the black-box and the preference-based optimization frameworks, which make different assumptions on the information available on the cost function. In the former case, the values assumed by the cost function can be measured by running computer simulations or performing real-world experiments. Instead, in the latter framework, the optimization is carried out using only the preferences expressed by a human decision-maker, who compares two tunings of the decision vector at a time and states which he/she likes the most. Surrogate-based methods can also handle black-box constraints (whose analytical formulations are unknown) by restricting the search only to those regions of the decision space that are likely to contain feasible calibrations. In particular, in the context of preference-based optimization, we have also highlighted a peculiar type of black-box constraints: decision-maker-based constraints. The latter can be seen as asking the decision-maker a "yes/no question" (such as "is this tuning acceptable?").

Contributions

Overall, we have covered five optimization frameworks: global optimization (Chapter 1), black-box optimization (unconstrained and constrained, Chapter 2) and preference-based optimization (unconstrained and constrained, Chapter 3). The first (minor) contribution of this book is the unified dissertation on the aforementioned frameworks that, in the end, all aim to solve a global optimization problem. Notably, we have shown that, by leveraging some results taken from the utility theory literature [104], the task of finding the calibration(s) that is (are) the most preferred by a human decision-maker is equivalent to solving a more traditional optimization problem. Hence, key results from the GO literature, such as the convergence theorem in [143] (i.e. Theorem 1.2), can also be applied to PBO.

The second contribution of this book is the derivation of globally convergent extensions for two recent surrogate-based methods: `GLIS` [10] (unconstrained BBO) and `GLISp` [11] (unconstrained PBO), giving rise to `GLIS-r` [108] and `GLISp-r` [109] (Chapter 5). We have addressed the shortcomings of the exploration function used by the original methods through the definition of a revisited infill sampling criterion. Moreover, we have proposed the greedy δ-cycling strategy, a simple yet effective way to cycle between exploiting the surrogate models and exploring the feasible region of the optimization problem. To the best of our knowledge, `GLISp-r` [109] is the first preference-based surrogate-based method with a formal proof of convergence. Empirically, in Chapter 7, we have also highlighted the robustness of `GLISp-r` [109] on several benchmark optimization problems, when compared to the original `GLISp` [11] algorithm, which is achieved without particularly compromising its convergence speed.

The third contribution of this book is the definition of a general unified surrogate-based scheme for unconstrained black-box and preference-based optimization, namely `gMRS` [108] (Chapter 5). Its global convergence is guaranteed for a wide variety of surrogate models of the cost function and exploration functions. Moreover, we have shown that `GLIS-r` [108] and `GLISp-r` [109] are none other than two specific implementations of the `gMRS` [108] scheme.

The fourth contribution of this book is the extension of `GLIS-r` [108] and `GLISp-r` [109] to the constrained black-box and preference-based optimization frameworks (Chapter 6), giving rise to `C-GLIS-r` and `C-GLISp-r`. To do so, we have proposed a slight modification to the probabilistic support vector machine classifier [15, 106], tailored specifically for constrained BBO and PBO. Moreover, we have derived a novel infill sampling criterion, which combines the probability of feasibility returned by the PSVM classifier with a slack variable. Its goal is to drive the search towards those regions of the decision space where the black-box constraints are likely to be satisfied but, at the same time, allow sampling promising calibrations that are on the other side of the decision boundary of the PSVM classifier. Empirically, in Chapter 7, we have shown that the proposed methods for constrained BBO and PBO are more sample efficient than `C-GLIS` and `C-GLISp` [156].

The fifth and last contribution of this book is the application of `GLIS-r` [108], `GLISp-r` [109], `C-GLIS-r` and `C-GLISp-r` to a control systems case study (Chapter 8). The proposed procedures have been employed to calibrate the PI + FF controllers of a hydraulic forming press. We have shown the usefulness of preference-based optimization methods when only qualitative control specifications are available, allowing us to skip the definition of performance indicators entirely. Moreover, we have highlighted the advantages of employing constrained BBO and PBO methods over the unconstrained ones: fewer unsatisfactory calibrations have been tried and we have been able to find good tunings using fewer sample evaluations. Lastly, decision-maker-based constraints have proven to be a powerful tool when performance requirements are hard to define.

Future work

The work presented in this book can be extended in several ways:

1. First of all, we would like to analyze the performances of the `gMRS` [108] scheme for several different surrogate models and exploration functions (see Sections 2.5, 3.4 and 5.4.2).

2. Moreover, it could be interesting to test the proposed black-box optimization methods in the presence of noisy measurements (i.e. when Assumption 2.4 does not hold).

3. Furthermore, in the preference-based setting, a compelling line of research could be the analysis of the robustness of `GLISp-r` [109] and `C-GLISp-r` in the presence of inconsistent decision-makers. From a theoretical perspective, inconsistent decision-makers could be seen as "irrational" (in a sense that the conditions in Definition 3.1 do not hold). In the utility theory literature, there already exist models for preference relations that are not complete and/or transitive, see [99, 103]. However, to the best of our knowledge, there are no preference-based optimization methods that explicitly handle decision-makers that are not rational. From a practical perspective, we would need a surrogate model that is able to handle the answer "I do not know" when the DM cannot decide which, among two calibrations, he/she prefers.

4. Similarly to the previous line of research, we would also like to delve into the topic of just noticeable differences (see Example 3.2). In particular, it could be interesting to derive infill sampling criteria that take such limitation into account. For example, it might be beneficial to propose new candidate samples that are at least ϵ-away from the previously evaluated ones, making it possible for the decision-maker to always distinguish between the calibrations that he/she is asked to compare.

5. In this book, we have only dealt with continuous decision variables. However, many engineering applications also include categorical and integer parameters that need to be tuned, see [78]. Therefore, future extensions of the proposed algorithms should also address mixed-variable optimization problems (see for example [17, 48]).

6. Last but not least, a future line of research is devoted to the derivation of a trust-region surrogate-based method for preference-based optimization. Roughly speaking, instead of solving a global optimization problem when proposing new candidate samples (see for example Section 5.2 and Section 6.2), we focus only in a neighborhood of the current best solution. In the context of black-box optimization, trust-region methods already exist, see [117, 152]. However, such

rationale has yet to be employed for preference-based optimization. Intuitively, a trust-region approach could be particularly useful when bounds on the decision variables are hard to define (such as in the case study presented in Chapter 8) or when a good initial calibration is already available.

Appendices

Appendix A. Mathematical background

This Appendix is devoted to reviewing some mathematical concepts that are needed for this book. The treatment is by no means complete, but it serves mainly to remind us of some key notions and to set out our notation. In particular, we cover:

- Appendix A.1: binary relations and, in particular, notions of order theory that are required to define the preference-based optimization problem. See [104, 127].
- Appendix A.2: metric spaces and their properties. Compact and dense subsets of metric spaces play a key role in global optimization and are also relevant for the definition of the utility function of a human decision-maker. The continuity of functions in metric spaces is also addressed. See [71, 104, 127].
- Appendix A.3: differentiability theory for multivariable functions, which is relevant from an optimization perspective. See [121, 130].
- Appendix A.4: basic optimization concepts related to unconstrained and constrained optimization problems. See [18, 100].

A.1 Order theory

As the name suggests, order theory is a branch of mathematics that provides the notion of order, making it possible to compare objects (elements) that pertain to the same category (set). This is closely related to preference-based optimization, wherein a decision-maker ranks several calibrations based on his/her personal tastes. Whenever the decision-maker states "calibration $\boldsymbol{x}_i$ is better than tuning $\boldsymbol{x}_j$", he/she is actually ranking $\boldsymbol{x}_i$ higher than $\boldsymbol{x}_j$, effectively imposing an order between the two.

The building blocks of order theory are ordered pairs. An ordered pair is an ordered list (a, b) consisting of two objects, a and b. It is ordered in a sense that, given two ordered pairs (a, b) and (c, d), we have $(a, b) = (c, d)$ if and only if $a = c$ and $b = d$.

A set of ordered pairs can be obtained by taking the Cartesian product of two nonempty sets $\mathcal{A}$ and $\mathcal{B}$:

$$\mathcal{A} \times \mathcal{B} = \{(a, b) : a \in \mathcal{A} \text{ and } b \in \mathcal{B}\} . \tag{A.1}$$

In order theory, we use binary relations to connect the elements of $\mathcal{A}$ to the members of $\mathcal{B}$.

Definition A.1: Binary relation. *Let $\mathcal{A}$ and $\mathcal{B}$ be two nonempty sets. A subset $\mathcal{R}$ of $\mathcal{A} \times \mathcal{B}$ is called a (binary) relation from $\mathcal{A}$ to $\mathcal{B}$ ($\mathcal{R} \subseteq \mathcal{A} \times \mathcal{B}$). If $\mathcal{A} = \mathcal{B}$, then we say that $\mathcal{R}$ is a (binary) relation on $\mathcal{A}$ ($\mathcal{R} \subseteq \mathcal{A} \times \mathcal{A}$).*

Notation and conventions. We denote the ordered pair $(a, b) \in \mathcal{R}$ as $a\mathcal{R}b$.

Relations can exhibit several different properties.

Definition A.2: Properties of relations. *A relation $\mathcal{R}$ on a nonempty set $\mathcal{A}$ is said to be:*

- *Reflexive if $a\mathcal{R}a$ for each $a \in \mathcal{A}$ (otherwise, it is said to be irreflexive);*
- *Complete if either $a\mathcal{R}b$ or $b\mathcal{R}a$ holds for each $a, b \in \mathcal{A}$;*
- *Symmetric if, for any $a, b \in \mathcal{A}$, $a\mathcal{R}b$ implies $b\mathcal{R}a$;*
- *Asymmetric if, for any $a, b \in \mathcal{A}$, $a\mathcal{R}b$ implies that $b\mathcal{R}a$ does not hold (and vice-versa);*
- *Antisymmetric if, for any $a, b \in \mathcal{A}$, $a\mathcal{R}b$ and $b\mathcal{R}a$ hold then $a = b$;*
- *Transitive if $a\mathcal{R}b$ and $b\mathcal{R}c$ imply $a\mathcal{R}c$ for any $a, b, c \in \mathcal{A}$.*

Next, we give some remarks on the just defined properties of relations.

Remark A.1. *It holds that:*

- *Every complete relation is reflexive;*
- *An asymmetric relation is a relation that is both antisymmetric and irreflexive;*

Now, we define a particular relation called the equivalence relation.

Definition A.3: Equivalence relation. *A relation $\sim$ on a nonempty set $\mathcal{A}$ is called an equivalence relation if it is reflexive, symmetric and transitive.*

Example A.1: Equivalence relation

Consider the set of real numbers $\mathbb{R}$ and any three of its elements $a, b, c \in \mathbb{R}$. The relation "is equal to", denoted as $=$, is an equivalence relation on $\mathbb{R}$ since the following properties hold:

- $a = a$ (reflexivity),
- If $a = b$ then $b = a$ (symmetry),
- If $a = b$ and $b = c$ then $a = c$ (transitivity).

Vice-versa, the relation "is greater than or equal to", denoted as $\geq$, is not an equivalence relation on $\mathbb{R}$ because it is not symmetric, i.e. if $a \geq b$ then it is not necessarily true that $b \geq a$. In particular, we have that $a \geq b \Rightarrow b \geq a$ if and only if $a = b$; thus, $\geq$ is an antisymmetric relation on $\mathbb{R}$.

In practice, any transitive relation $\mathcal{R}$ on a set $\mathcal{A}$ can be used to define an order between some (but not necessarily all) the elements of $\mathcal{A}$. The relation $\mathcal{R}$ on $\mathcal{A}$ assumes different names based on which of the properties in Definition A.2 hold.

Definition A.4: Order relations and ordered sets. *A relation $\mathcal{R}$ on a nonempty set $\mathcal{A}$ is called:*

- *A preorder on $\mathcal{A}$ if it is transitive and reflexive,*
- *A partial order on $\mathcal{A}$ if it is an antisymmetric preorder on $\mathcal{A}$,*
- *A linear order on $\mathcal{A}$ if it is a partial order on $\mathcal{A}$ which is complete.*

Furthermore, $(\mathcal{A}, \mathcal{R})$ is said to be:

- *A preordered set if $\mathcal{R}$ is a preorder on $\mathcal{A}$,*
- *A poset (partially ordered set) if $\mathcal{R}$ is a partial order on $\mathcal{A}$,*
- *A loset (linearly ordered set) if $\mathcal{R}$ is a linear order on $\mathcal{A}$.*

For the remainder of this Appendix, as well as for preference-based optimization, we use a more expressive notation for preorders.

Notation and conventions. From now on, we will always refer to a generic preorder as $\gtrsim$ instead of $\mathcal{R}$. Furthermore, we denote the asymmetric part of $\gtrsim$ as $\succ$, and the symmetric part of $\gtrsim$ as $\sim$.

Partially ordered sets play a key role in mathematics. Intuitively, if $(\mathcal{A}, \gtrsim)$ is a poset, then some of the elements of $\mathcal{A}$ can be ordered in some sense (according to $\gtrsim$). Therefore, we can define lower bounds, upper bounds, suprema, infima, maxima and minima of partially ordered sets as follows.

Definition A.5: $\gtrsim$-upper bound and $\gtrsim$-lower bound of a poset. *Let $(\mathcal{A}, \gtrsim)$ be a poset and $\mathcal{B} \subseteq \mathcal{A}$. An element $a \in \mathcal{A}$ is said to be:*

- *A $\gtrsim$-upper bound of $\mathcal{B}$ if $a \gtrsim b, \forall b \in \mathcal{B}$. We then say that $\mathcal{B}$ is bounded above.*
- *A $\gtrsim$-lower bound of $\mathcal{B}$ if $b \gtrsim a, \forall b \in \mathcal{B}$. We then say that $\mathcal{B}$ is bounded below.*

Furthermore, we say that a set is (order) bounded if and only if it is bounded both above and below.

Definition A.6: $\succsim$-supremum and $\succsim$-infimum of a poset. *Let $(\mathcal{A}, \succsim)$ be a poset and $\mathcal{B} \subseteq \mathcal{A}$. An element $a \in \mathcal{A}$ is said to be:*

- *A $\succsim$-supremum of $\mathcal{B}$ if it is a $\succsim$-upper bound of $\mathcal{B}$ such that, for all $\succsim$-upper bounds $\tilde{a} \in \mathcal{A}$ of $\mathcal{B}$, $\tilde{a} \succsim a$. For this reason, the $\succsim$-supremum is often referred to as the least $\succsim$-upper bound. We denote it as $\sup_{\succsim} \mathcal{B}$.*
- *A $\succsim$-infimum of $\mathcal{B}$ if it is a $\succsim$-lower bound of $\mathcal{B}$ such that, for all $\succsim$-lower bounds $\tilde{a} \in \mathcal{A}$ of $\mathcal{B}$, $a \succsim \tilde{a}$. For this reason, the $\succsim$-infimum is often referred to as the greatest $\succsim$-lower bound. We denote it as $\inf_{\succsim} \mathcal{B}$.*

Note that there can be only one $\succsim$-supremum and only one $\succsim$-infimum of any subset of $\mathcal{A}$.

Definition A.7: $\succsim$-maximum and $\succsim$-minimum of a poset. *Let $(\mathcal{A}, \succsim)$ be a poset and $\emptyset \neq \mathcal{B} \subseteq \mathcal{A}$. An element $a \in \mathcal{B}$ is said to be:*

- *A $\succsim$-maximum of $\mathcal{B}$, denoted as $\max_{\succsim} \mathcal{B}$, if $a \succsim b$ for all $b \in \mathcal{B}$,*
- *A $\succsim$-minimum of $\mathcal{B}$, denoted as $\min_{\succsim} \mathcal{B}$, if $b \succsim a$ for all $b \in \mathcal{B}$.*

Note that any nonempty subset of a poset can have at most one $\succsim$-maximum and one $\succsim$-minimum.

We now give some remarks on the lower bounds, upper bounds, supremum, infimum, maximum and minimum of a subset $\mathcal{B}$ of a poset $(\mathcal{A}, \succsim)$.

Remark A.2. *We stress that:*

- *The $\succsim$-supremum and the $\succsim$-infimum of $\mathcal{B}$, if they exist, need not be inside $\mathcal{B}$. Instead, the $\succsim$-maximum and the $\succsim$-minimum of $\mathcal{B}$, if they exist, must be elements of $\mathcal{B}$.*
- *$a = \max_{\succsim} \mathcal{B}$ if and only if $a \in \mathcal{B}$ and a is a $\succsim$-upper bound of $\mathcal{B}$.*
- *$a = \min_{\succsim} \mathcal{B}$ if and only if $a \in \mathcal{B}$ and a is a $\succsim$-lower bound of $\mathcal{B}$.*
- *If $\max_{\succsim} \mathcal{B}$ exists, then $\max_{\succsim} \mathcal{B} = \sup_{\succsim} \mathcal{B}$.*
- *If $\min_{\succsim} \mathcal{B}$ exists, then $\min_{\succsim} \mathcal{B} = \inf_{\succsim} \mathcal{B}$.*
- *$\mathcal{A}$ is (order) bounded if and only if it has both a maximum and a minimum.*

Definition A.7 is particularly relevant for the derivation of the preference-based optimization problem. In this context, $\gtrsim$ is a relation that describes the tastes of a human decision-maker. The objective of a preference-based optimization procedure is to find the $\gtrsim$-maximum of some subset Ω of $\mathbb{R}^n$, i.e. to find the element of Ω that is most preferred by the decision-maker.

The next Example gives us an order theoretic perspective on Euclidean spaces.

Example A.2: Natural order of Euclidean spaces

Consider the n-dimensional Euclidean space $\mathbb{R}^n, n \in \mathbb{N}$. It is easy to see that:

- $(\mathbb{R}^n, \geq)$ is a poset. The "greater than or equal to" relation $\geq$ on $\mathbb{R}^n$ is defined component-wise, i.e. given $\boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathbb{R}^n$, we say that $\boldsymbol{x}_1 \geq \boldsymbol{x}_2$ if and only if $x_1^{(i)} \geq x_2^{(i)}, \forall i = 1, \dots, n$;
- $(\mathbb{R}, \geq)$ is a loset.

Typically, whenever we talk about $\mathbb{R}^n$, we implicitly assume that it is ordered by $\geq$ (for this reason, $\geq$ is often referred to as the *natural order of* $\mathbb{R}^n$). Furthermore, given any subset Ω of $\mathbb{R}^n$, we forego $\geq$ in the notation of the bounds, supremum, infimum, maximum and minimum. Therefore, we simply write $\sup \Omega$ instead of $\sup_{\geq} \Omega$ and similarly for all the other operators.

A.1.1 Functions as binary relations

Conceptually, a function can be seen as an operation that transforms inputs into outputs and is none other than a binary relation. In particular, a function f that maps $\mathcal{A}$ into $\mathcal{B}$, denoted as $f : \mathcal{A} \to \mathcal{B}$, is a relation $f \subseteq \mathcal{A} \times \mathcal{B}$ such that:

- For every $a \in \mathcal{A}$, there exists a $b \in \mathcal{B}$ such that afb holds,
- For every $b_1, b_2 \in \mathcal{B}$ and $a \in \mathcal{A}$ such that afb_1 and afb_2, we have $b_1 = b_2$ (i.e. an element of $\mathcal{A}$ has associated one and only one element of $\mathcal{B}$).

Furthermore, $\mathcal{A}$ is called the domain of f and $\mathcal{B}$ is the codomain of f. Given any set $C \subseteq \mathcal{A}$ and a function $f : \mathcal{A} \to \mathcal{B}$, it is convenient to introduce the following notation:

$$f[C] = \{b : b \in \mathcal{B}, cfb \text{ for some } c \in C\}. \tag{A.2}$$

Then, the range of $f : \mathcal{A} \to \mathcal{B}$ is defined as:

$$\operatorname{range}(f) = f[\mathcal{A}]. \tag{A.3}$$

Notation and conventions. In practice, a more commonly used notation for afb is $f(a) = b$; b is referred to as the image (or value) of a under f. Furthermore, whenever we refer to the function "as a whole", for the sake of clarity we either use the notation $f(\cdot)$ or $f(a)$ instead of simply f.

A.2 Metric spaces

Many properties of multivariable functions defined over the n-dimensional Euclidean space, i.e. such that $f : \mathbb{R}^n \to \mathbb{R}$, depend on the notion of distance between points in $\mathbb{R}^n$. For example, intuitively speaking, continuity of $f(\boldsymbol{x})$ implies that if we take two points $\boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathbb{R}^n$ that are "close" to each other (i.e. the distance between the two is small), then the images $f(\boldsymbol{x}_1), f(\boldsymbol{x}_2)$ must also be "close". Formally, the n-dimensional Euclidean space is a metric space, i.e. a set for which it is possible to define a distance between any two of its elements. In general, metric spaces are defined as follows.

Definition A.8: Metric spaces. *Let $\mathcal{A}$ be any nonempty set. A function $d : \mathcal{A} \times \mathcal{A} \to \mathbb{R}_{\geq 0}$ that satisfies the following properties for any $a, b, c \in \mathcal{A}$:*

1. *$d(a, b) = 0$ if and only if $a = b$,*
2. *$d(a, b) = d(b, a)$ (symmetry),*
3. *$d(a, c) \leq d(a, b) + d(b, c)$ (triangle inequality)*

is called a distance function (or metric) on $\mathcal{A}$. If $d(\cdot, \cdot)$ is a distance function on $\mathcal{A}$, we say that $(\mathcal{A}, d)$ is a metric space, and refer to the elements of $\mathcal{A}$ as points in $(\mathcal{A}, d)$.

Notation and conventions. Whenever the metric under consideration is apparent from the context, it is customary to dispense the notation $(\mathcal{A}, d)$ and refer to $\mathcal{A}$ as a metric space.

If $\mathcal{A}$ is a metric space (with metric $d(\cdot, \cdot)$) and $\emptyset \neq \mathcal{B} \subset \mathcal{A}$, we can view $\mathcal{B}$ as a metric space in its own right by using the distance function induced by $d(\cdot, \cdot)$ on $\mathcal{B}$. More precisely, we make $\mathcal{B}$ a metric space by means of the distance function $d' : \mathcal{B} \times \mathcal{B} \to \mathbb{R}_{\geq 0}$, $d'(a, b) = d(a, b), \forall a, b \in \mathcal{B}$. We then say that $(\mathcal{B}, d')$, or simply $\mathcal{B}$, is a metric subspace of $\mathcal{A}$.

The next Example shows some trivial metric spaces.

Example A.3: Discrete and Euclidean spaces

Some examples of metric spaces are:

- Let $\mathcal{A}$ be any nonempty set. A trivial way of making $\mathcal{A}$ a metric space is to use the metric $d : \mathcal{A} \times \mathcal{A} \to \mathbb{R}_{\geq 0}$ defined as:

$$d(a,b) = \begin{cases} 1 & \text{if } a \neq b \\ 0 & \text{if } a = b \end{cases}.$$

  $(\mathcal{A}, d)$ is a metric space called discrete space and $d(\cdot,\cdot)$ is referred to as the discrete metric on $\mathcal{A}$.

- $(\mathbb{R}^n, d_p)$ is a metric space for each $1 \leq p \leq \infty$, where $d_p : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}_{\geq 0}$ is defined as:

$$d_p(\boldsymbol{x}_1, \boldsymbol{x}_2) = \left(\sum_{i=1}^{n} \left| x_1^{(i)} - x_2^{(i)} \right|^p \right)^{\frac{1}{p}} \qquad \text{for } 1 \leq p < \infty, \tag{A.4}$$

$$d_p(\boldsymbol{x}_1, \boldsymbol{x}_2) = \max_{i \in \{1,\dots,n\}} \left| x_1^{(i)} - x_2^{(i)} \right| \qquad \text{for } p = \infty. \tag{A.5}$$

  In particular, $(\mathbb{R}^n, d_2)$ is called the n-dimensional Euclidean space and is often denoted simply as $\mathbb{R}^n$ (implicitly metrized by $d_2(\cdot,\cdot)$).

To be more precise, the n-dimensional Euclidean space is a normed vector space. Normed vector spaces are subsets of metric spaces on which a norm is defined, as we will briefly review in the next Appendix.

A.2.1 Normed vector spaces

Before introducing normed vector spaces, we need to define vector spaces.

Definition A.9: Vector spaces. *A vector space over a field $\mathcal{F}$ is a set $\mathcal{V}$ endowed with two operations:*

1. *Vector addition $+ : \mathcal{V} \times \mathcal{V} \to \mathcal{V}$, i.e. $v + w \in \mathcal{V}, \forall v, w \in \mathcal{V}$;*
2. *Scalar multiplication $\cdot : \mathcal{F} \times \mathcal{V} \to \mathcal{V}$, i.e. $\lambda \cdot v \in \mathcal{V}, \forall v \in \mathcal{V}, \forall \lambda \in \mathcal{F}$.*

The elements of $\mathcal{V}$ are referred to as vectors, while the elements of $\mathcal{F}$ are called scalars. We denote the vector space as $(\mathcal{V}, \mathcal{F})$.
Vector addition and scalar multiplication must satisfy the following axioms:

- *$v + w = w + v, \forall v, w \in \mathcal{V}$ (additive commutativity);*
- *$(u + v) + w = u + (v + w), \forall u, v, w \in \mathcal{V}$ (additive associativity);*

- *$\exists 0 \in \mathcal{V} : v + 0 = v, \forall v \in \mathcal{V}$ (identity element of vector addition);*
- *For every $v \in \mathcal{V}$ there exists a $w \in \mathcal{V}$ such that $v + w = 0$ (existence of the additive inverse). We denote w as $-v$;*
- *$(\lambda_1 + \lambda_2) \cdot v = \lambda_1 \cdot v + \lambda_2 \cdot v, \forall v \in \mathcal{V}, \forall \lambda_1, \lambda_2 \in \mathcal{F}$ (distributivity of scalar multiplication with respect to scalar addition);*
- *$\lambda \cdot (v + w) = \lambda \cdot v + \lambda \cdot w, \forall v, w \in \mathcal{V}, \forall \lambda \in \mathcal{F}$ (distributivity of scalar multiplication with respect to vector addition);*
- *$(\lambda_1 \cdot \lambda_2) \cdot v = \lambda_1 \cdot (\lambda_2 \cdot v), \forall v \in \mathcal{V}, \forall \lambda_1, \lambda_2 \in \mathcal{F}$ (multiplicative associativity);*
- *$\exists 1 \in \mathcal{F} : 1 \cdot v = v, \forall v \in \mathcal{V}$ (identity element of scalar multiplication).*

From the previous Definition, $\mathcal{F}$ can be an arbitrary field of scalars. Often, $\mathcal{F} = \mathbb{R}$ or $\mathcal{F} = \mathbb{C}$ and we refer to $(\mathcal{V}, \mathbb{R})$ and $(\mathcal{V}, \mathbb{C})$ as a real vector space and a complex vector space respectively.

Remark A.3 (Metric spaces and vector spaces)**.** *A vector space with no additional structure is not necessarily a metric space and vice-versa. For instance, a metric space might have no notion of addition and, similarly, a vector space does not necessarily possess a metric.*

We can link vector spaces to metric spaces by enriching the former with a norm, giving rise to normed vector spaces.

Definition A.10: Normed vector spaces. *Consider a real or complex vector space $(\mathcal{V}, \mathcal{F})$ ($\mathcal{F} = \mathbb{R}$ or $\mathcal{F} = \mathbb{C}$). A norm over $\mathcal{V}$ is a function $\|\cdot\| : \mathcal{V} \to \mathbb{R}_{\geq 0}$ that satisfies the following properties for any $v, w \in \mathcal{V}, \lambda \in \mathcal{F}$:*

1. *$\|v\| = 0$ if and only if $v = 0$,*
2. *$\|\lambda \cdot v\| = |\lambda| \cdot \|v\|$,*
3. *$\|v + w\| \leq \|v\| + \|w\|$ (triangle inequality).*

In this case, we say that $(\mathcal{V}, \|\cdot\|)$ is a normed vector space.

The next Remark shows the relationship between normed vector spaces and metric spaces.

Remark A.4 (Metric spaces and normed vector spaces)**.** *By enriching a real or complex vector space with a norm, giving rise to the normed vector space $(\mathcal{V}, \|\cdot\|)$, we make $\mathcal{V}$ a metric space. That is because the norm $\|\cdot\|$ induces the distance $d : \mathcal{V} \times \mathcal{V} \to \mathbb{R}_{\geq 0}$, called the (norm) induced metric over*

$\mathcal{V}$, defined as:

$$d(v, w) = \|v - w\|.$$

Then, $(\mathcal{V}, d)$ with $d(\cdot,\cdot)$ defined as above is a metric space.

Example A.4: Norms for $\mathcal{V} = \mathbb{R}^n$

Consider the normed vector space $(\mathbb{R}^n, \|\cdot\|)$. The distances proposed in (A.4) and (A.5) are (norm) induced metrics derived from the p-norm:

$$\|\boldsymbol{x}\|_p = \left(\sum_{i=1}^{n} \left|x^{(i)}\right|^p\right)^{\frac{1}{p}} \quad \text{for } 1 \le p < \infty \tag{A.6}$$

and max norm:

$$\|\boldsymbol{x}\|_\infty = \max_{i=1,\dots,n} \left|x^{(i)}\right| \tag{A.7}$$

respectively. Moreover, the 2-norm is referred to as the Euclidean norm and can be written as:

$$\|\boldsymbol{x}\|_2 = \sqrt{\sum_{i=1}^{n} \left(x^{(i)}\right)^2}. \tag{A.8}$$

We can view the n-dimensional Euclidean space as:

- A normed vector space $(\mathbb{R}^n, \|\cdot\|_2)$,
- A metric space $(\mathbb{R}^n, d_2)$, where $d_2(\boldsymbol{x}_1, \boldsymbol{x}_2) = \|\boldsymbol{x}_1 - \boldsymbol{x}_2\|_2$ is the metric induced by the Euclidean norm.

A.2.2 Sequences in metric spaces

Some of the concepts related to metric spaces that we are about to see are connected to the notion of sequences. Let $(\mathcal{A}, d)$ be a nonempty metric space, we define a sequence in $\mathcal{A}$ as a function $h : \mathbb{N} \to \mathcal{A}$, represented as

$$\langle a_i \rangle_{i \ge 1} = \langle a_1, a_2, \dots \rangle,$$

such that $a_i = h(i)$ for each $i \in \mathbb{N}$. In practice, a sequence is an ordered list of elements of $\mathcal{A}$. By a subsequence of a sequence, we mean a sequence that is made up of the terms of $\langle a_i \rangle_{i \ge 1}$ which appear in the subsequence in the same order as they appear in $\langle a_i \rangle_{i \ge 1}$. For example, we define the subsequence of $\langle a_i \rangle_{i \ge 1}$ composed of its first $k \in \mathbb{N}$ elements as:

$$\langle a_i \rangle_{i=1}^{k} = \langle a_1, \dots, a_k \rangle.$$

A sequence is said to be finite if it has limited number of terms and infinite if it does not.

Convergent sequences are defined as follows.

Definition A.11: Convergent sequences. *Let $(\mathcal{A}, d)$ be a nonempty metric space and let $\langle a_i \rangle_{i\geq 1}$ be a sequence in $\mathcal{A}$. We say that $\langle a_i \rangle_{i\geq 1}$ converges to $\tilde{a} \in \mathcal{A}$ if, for each $\epsilon \in \mathbb{R}_{>0}$, there exists an index $K \in \mathbb{N}$ (that may depend on ϵ) such that:*

$$d(a_k, \tilde{a}) < \epsilon, \quad \forall k \geq K.$$

In this case, we say that $\langle a_i \rangle_{i\geq 1}$ is convergent in $\mathcal{A}$. $\tilde{a}$ is said to be the limit of $\langle a_i \rangle_{i\geq 1}$ and it is denoted as:

$$\lim_{i\to\infty} a_i = \tilde{a}.$$

If $\langle a_i \rangle_{i\geq 1}$ is not convergent in $\mathcal{A}$, then it is said to be divergent in $\mathcal{A}$.

Remark A.5. *Any sequence in a metric space can converge to at most one limit.*

A.2.3 Compactness and denseness of metric spaces

In this Appendix, we introduce some key properties for metric spaces, which are quite relevant from an optimization perspective. We begin by defining an important metric subspace, namely the neighborhood of a point in a metric space.

Definition A.12: ϵ-neighborhood and neighborhood of a point. *Let $(\mathcal{A}, d)$ be a metric space. For any $a \in \mathcal{A}$ and $\epsilon \in \mathbb{R}_{>0}$, we define the ϵ-neighborhood of a as the set:*

$$\mathcal{N}_{\mathcal{A}}(a; \epsilon) = \{b : b \in \mathcal{A}, d(a, b) < \epsilon\}. \tag{A.9}$$

In turn, a neighborhood of a in $\mathcal{A}$, denoted as $\mathcal{N}_{\mathcal{A}}(a)$, is any subset of $\mathcal{A}$ that contains at least one ϵ-neighborhood of $a \in \mathcal{A}$.

Remark A.6. *The ϵ-neighborhood of a point a in a metric space $\mathcal{A}$ is never empty, it contains at least a.*

Notation and conventions. Usually, when it is clear from the context, the dependency on the metric space $\mathcal{A}$ is omitted and an ϵ-neighborhood of a is simply denoted as $\mathcal{N}(a; \epsilon)$. Similarly for a neighborhood of a.

ϵ-neighborhoods are used to define open and closed sets as follows.

Definition A.13: Open and closed subsets of a metric space. *Let $\mathcal{A}$ be a metric space. A subset $\mathcal{B}$ of $\mathcal{A}$ is said to be open in $\mathcal{A}$ (or an open subset of $\mathcal{A}$) if, for each $b \in \mathcal{B}$, there exists an $\epsilon \in \mathbb{R}_{>0}$ such that $\mathcal{N}_{\mathcal{A}}(b; \epsilon) \subseteq \mathcal{B}$.*
A subset $\mathcal{B}$ of $\mathcal{A}$ is said to be closed in $\mathcal{A}$ (or a closed subset of $\mathcal{A}$) if $\mathcal{A} \setminus \mathcal{B}$ is open in $\mathcal{A}$.
There also exist sets that are both open and closed, which are called clopen.

Alternatively, closed sets can be characterized using sequences.

Proposition A.1. *A subset $\mathcal{B}$ of a metric space $\mathcal{A}$ is closed if and only if every sequence in $\mathcal{B}$ that is convergent in $\mathcal{A}$ converges to a point in $\mathcal{B}$.*

The most commonly used neighborhoods for $\mathbb{R}^n$ are open balls, as described in the following Example.

Example A.5: Open and closed balls

Consider the n-dimensional Euclidean space $\mathbb{R}^n$. We define the open ball of radius $\epsilon \in \mathbb{R}_{>0}$ around $\tilde{\boldsymbol{x}} \in \mathbb{R}^n$ as:

$$\mathcal{B}(\tilde{\boldsymbol{x}}; \epsilon) = \{\boldsymbol{x} : \|\boldsymbol{x} - \tilde{\boldsymbol{x}}\|_2 < \epsilon\}. \tag{A.10}$$

As the name suggest, an open ball is an open subset of $\mathbb{R}^n$. Conversely, the closed ball is defined as:

$$\bar{\mathcal{B}}(\tilde{\boldsymbol{x}}; \epsilon) = \{\boldsymbol{x} : \|\boldsymbol{x} - \tilde{\boldsymbol{x}}\|_2 \leq \epsilon\}. \tag{A.11}$$

In this case, $\bar{\mathcal{B}}(\tilde{\boldsymbol{x}}, \epsilon)$ is a closed subset of $\mathbb{R}^n$.

Some important open and closed sets are the following.

Definition A.14: Interior, closure and boundary. *Let $\mathcal{A}$ be a metric space and $\mathcal{B} \subseteq \mathcal{A}$. Then:*

- *The largest open set in $\mathcal{A}$ that is contained in $\mathcal{B}$ is called the interior of $\mathcal{B}$ (relative to $\mathcal{A}$) and is denoted by $int_{\mathcal{A}}(\mathcal{B})$;*
- *The smallest closed set in $\mathcal{A}$ that contains $\mathcal{B}$ is called the closure of $\mathcal{B}$ (relative to $\mathcal{A}$) and is denoted by $cl_{\mathcal{A}}(\mathcal{B})$;*
- *The boundary of $\mathcal{B}$ (relative to $\mathcal{A}$), denoted by $bd_{\mathcal{A}}(\mathcal{B})$, is defined as:*

$$bd_{\mathcal{A}}(\mathcal{B}) = cl_{\mathcal{A}}(\mathcal{B}) \setminus int_{\mathcal{A}}(\mathcal{B}).$$

In Appendix A.1, Definition A.5, we have already seen the notion of boundedness for partially ordered sets, which depends on the preorder $\gtrsim$. In metric spaces, we can also define what it means for a metric subspace to be bounded, this time relying on the distance $d(\cdot,\cdot)$ instead of the relation $\gtrsim$.

Definition A.15: Bounded subsets of a metric space. *A subset $\mathcal{B}$ of a metric space $\mathcal{A}$ is called (metrically) bounded (in $\mathcal{A}$) if there exists an $\epsilon \in \mathbb{R}_{>0}$ such that $\mathcal{B} \subseteq \mathcal{N}_{\mathcal{A}}(b;\epsilon)$ for some $b \in \mathcal{B}$. If $\mathcal{B}$ is not bounded, then it is said to be unbounded.*

Remark A.7 (Boundedness in metric spaces and in order theory)**.** *In general, the concepts of order-boundedness (i.e. for partially ordered sets) and metric-boundedness (i.e. for metric spaces) are not necessarily linked.*

A particular case is the set $\mathbb{R}$, which can be seen either as a loset $(\mathbb{R},\geq)$ or as a metric space $(\mathbb{R}, d_2(\cdot,\cdot))$. The following Proposition holds.

Proposition A.2: Order-boundedness and metric-boundedness in $\mathbb{R}$***.*** *A subset $\mathcal{A}$ of $\mathbb{R}$ is bounded with respect to the metric $d_2(\cdot,\cdot)$ in* (A.4) *if and only if it is order-bounded with respect to the preorder $\geq$. Therefore, the meanings of order-boundedness and metric-boundedness coincide, at least for subsets of $\mathbb{R}$.*

Next, we cover the concept of denseness of a subset of a metric space. Dense sets are particularly relevant for global optimization: a necessary condition for the global convergence of any optimization algorithm `A` is that, ultimately, `A` must produce a sequence of iterates that is dense with respect to the constraint set of the optimization problem.

Definition A.16: Dense subsets of a metric space. *Let $(\mathcal{A}, d)$ be a metric space and $\mathcal{B} \subseteq \mathcal{A}$. If $cl_{\mathcal{A}}(\mathcal{B}) = \mathcal{A}$, then $\mathcal{B}$ is said to be dense in $\mathcal{A}$ (or a dense subset of $\mathcal{A}$). Equivalently, $\mathcal{B}$ is dense in $\mathcal{A}$ if, for every $a \in \mathcal{A}$ and every $\epsilon \in \mathbb{R}_{>0}$, there exists a $b \in \mathcal{B}$ such that:*

$$d(a,b) < \epsilon.$$

The denseness of a subset of a metric space can also be assessed through sequences, as pointed out by the following Lemma.

Lemma A.1. *A subset $\mathcal{B}$ of a metric space $\mathcal{A}$, $\mathcal{B} \subseteq \mathcal{A}$, is dense if and only if for any $a \in \mathcal{A}$ there exists a sequence in $\mathcal{B}$, $\langle b_i \rangle_{i\geq 1}$, such that:*

$$\lim_{i\to\infty} b_i = a.$$

Now, we introduce the concept of compactness of a subset of a metric space. Compact sets also play a key role in optimization. In particular, as we will see later in this Appendix, any continuous function

$f : \mathbb{R}^n \to \mathbb{R}$ assumes a minimum and a maximum value over any compact subset of $\mathbb{R}^n$. In some sense, compactness provides a finite structure for infinite sets. In general, compact subsets of metric spaces rely on the notion of open cover.

Definition A.17: Open cover of a subset of a metric space. *Let $\mathcal{A}$ be a metric space and $\mathcal{B} \subseteq \mathcal{A}$. A class O of subsets of $\mathcal{A}$, i.e.*

$$O = \left\{ O^{(1)}, O^{(2)}, \dots \right\}, \quad O^{(i)} \subseteq \mathcal{A}, \forall O^{(i)} \in O,$$

is said to cover $\mathcal{B}$ if:

$$\mathcal{B} \subseteq \bigcup_{O^{(i)} \in O} O^{(i)}.$$

If all the members of O are open in $\mathcal{A}$, then we say that O is an open cover of $\mathcal{B}$.

Then, compact subsets of a metric space can be defined as follows.

Definition A.18: Compact subsets of a metric space. *A metric space $\mathcal{A}$ is said to be compact if every open cover of $\mathcal{A}$ has a finite subset that also covers $\mathcal{A}$. A subset $\mathcal{B}$ of $\mathcal{A}$ is said to be compact in $\mathcal{A}$ (or a compact subset of $\mathcal{A}$) if every open cover of $\mathcal{B}$ has a finite subset that also covers $\mathcal{A}$.*

Remark A.8. *Note that:*

- *If C is a compact subset of a metric space $\mathcal{B}$ and $\mathcal{B}$ is a metric subspace of $\mathcal{A}$, then C is compact in $\mathcal{A}$,*
- *A finite subset of any metric space is compact.*

Compactness of a metric space can also be assessed using sequences.

Theorem A.1: Sequential compactness in metric spaces

A subset $\mathcal{B}$ of a metric space $\mathcal{A}$ is compact if and only if every sequence in $\mathcal{B}$ has a subsequence that converges to a point in $\mathcal{B}$.

The next Propositions link the notions of closedness (Definition A.13) and (metric) boundedness (Definition A.15) of a subset of a metric space to its compactness.

Proposition A.3. *Any closed subset of a compact metric space is compact.*

Proposition A.4. *Any compact subset of a metric space is closed and bounded.*

It is important to note that not all closed and bounded subsets of a metric space are compact. In general, compactness is a stronger property than closedness and boundedness put together. An exception is the n-dimensional Euclidean space, for which the following Theorem holds.

Theorem A.2: Compactness of any subset of $\mathbb{R}^n$

Given any $n \in \mathbb{N}$, a subset of $\mathbb{R}^n$ is compact if and only if it is closed and bounded. This holds for the metric space $\left(\mathbb{R}^n, d_p\right)$ and any $1 \leq p \leq \infty$.

Any n-dimensional cube (or hypercube) is compact in $\mathbb{R}^n$, as claimed by the following Theorem.

Theorem A.3: Heine-Borel Theorem

For any $u, l \in \mathbb{R}, l \leq u$, the n-dimensional cube $\mathcal{A} = [l, u]^n$ is compact in $\mathbb{R}^n$.

By combining Theorem A.3 and Proposition A.3, we obtain the following Lemma.

Lemma A.2: Compactness of a box. *Any subset $\mathcal{A}$ of $\mathbb{R}^n$ that is defined as a box, i.e.*

$$\mathcal{A} = \{\boldsymbol{x} : \boldsymbol{l} \leq \boldsymbol{x} \leq \boldsymbol{u}\},$$

where $\boldsymbol{l}, \boldsymbol{u} \in \mathbb{R}^n, l^{(i)} \leq u^{(i)}, \forall i = 1, \dots, n$, or, alternatively,

$$\mathcal{A} = \left[l^{(1)}, u^{(1)}\right] \times \dots \times \left[l^{(n)}, u^{(n)}\right],$$

is a compact subset of $\mathbb{R}^n$.

The previous Lemma is particularly relevant for this book since the feasible regions of most black-box and preference-based optimization problems are boxes.

Lastly, the following Proposition gives us some insights on which set operations preserve compactness.

Proposition A.5: Set operations that preserve compactness. *Let $\mathcal{A}$ be a metric space and $\mathcal{B}$ and C be two compact subsets of $\mathcal{A}$. Then, the sets $\mathcal{B} \cap C$ and $\mathcal{B} \cup C$ are also compact.*

A.2.4 Continuity theory for metric spaces

In Appendix A.1, we have introduced functions as a particular case of binary relations. Here, we analyze the continuity property for those functions whose domains and codomains are metric spaces. Conceptually, if a function is continuous, then small perturbations on the inputs produce small perturbations on the outputs. More formally:

Definition A.19: Continuity of a function. *Let $(\mathcal{A}, d_{\mathcal{A}})$ and $(\mathcal{B}, d_{\mathcal{B}})$ be two metric spaces. We say that a function $f : \mathcal{A} \to \mathcal{B}$ is* <u>*continuous at $a_1 \in \mathcal{A}$*</u> *if, for any $\epsilon \in \mathbb{R}_{>0}$, there exists a $\delta \in \mathbb{R}_{>0}$ (which may depend on both ϵ and a_1) such that*

$$d_{\mathcal{A}}(a_1, a_2) < \delta \Rightarrow d_{\mathcal{B}}(f(a_1), f(a_2)) < \epsilon$$

for each $a_2 \in \mathcal{A}$, that is:

$$d_{\mathcal{B}}(f(a_1), f(a_2)) < \epsilon, \quad \forall a_2 \in \mathcal{N}_{\mathcal{A}}(a_1; \delta).$$

If $f(\cdot)$ is not continuous at a_1, then it is said to be <u>*discontinuous at a_1*</u>*.*
$f(\cdot)$ is said to be <u>*continuous*</u> *(or* <u>*continuous everywhere*</u>*) if it is continuous at each $a_1 \in \mathcal{A}$.*

Remark A.9. *The continuity of a function that maps a metric space to another depends intrinsically on the involved metrics. In particular, it depends on the distance functions $d_{\mathcal{A}}(\cdot,\cdot)$ and $d_{\mathcal{B}}(\cdot,\cdot)$ used to metrize $\mathcal{A}$ (domain) and $\mathcal{B}$ (codomain) respectively.*

The previous notion of continuity is, inherently, a local one. In practice, if $f : \mathcal{A} \to \mathcal{B}$ is continuous, we have that, for any $a \in \mathcal{A}$, the images of the points "nearby" a under $f(\cdot)$ are "close" to $f(a)$, but we do not know if the word "nearby" depends on a or not. A more global property is the following:

Definition A.20: Uniform continuity of a function. *Let $(\mathcal{A}, d_{\mathcal{A}})$ and $(\mathcal{B}, d_{\mathcal{B}})$ be two metric spaces. We say that a function $f : \mathcal{A} \to \mathcal{B}$ is* <u>*uniformly continuous*</u> *if, for all $\epsilon \in \mathbb{R}_{>0}$, there exists a $\delta \in \mathbb{R}_{>0}$ (which may depend on ϵ) such that*[a]*:*

$$f[\mathcal{N}_{\mathcal{A}}(a; \delta)] \subseteq \mathcal{N}_{\mathcal{B}}(f(a); \epsilon), \quad \forall a \in \mathcal{A}.$$

[a]Recall the notation in (A.2).

Intuitively speaking, if a function is uniformly continuous, then given any $\epsilon \in \mathbb{R}_{>0}$, it is possible to find a $\delta \in \mathbb{R}_{>0}$ such that, for any point $a \in \mathcal{A}$, the images of points at most δ-away from a under $f(\cdot)$ are at most ϵ-away from $f(a)$. Therefore, uniform continuity says something about the behavior of $f(\cdot)$ on its entire domain. It is easy to see that any uniformly continuous function is continuous, but the opposite does not necessarily hold.

Some other kinds of continuity that demand more regularity from a function are defined as follows.

Definition A.21: Other kinds of continuity. *Let $(\mathcal{A}, d_{\mathcal{A}})$ and $(\mathcal{B}, d_{\mathcal{B}})$ be two metric spaces. A function $f : \mathcal{A} \to \mathcal{B}$ is said to be:*

- *α-Hölder continuous, where $\alpha \in \mathbb{R}_{>0}$ is called the exponent of the Hölder condition, if there exists a constant $C \in \mathbb{R}_{>0}$ such that:*

$$d_{\mathcal{B}}\left(f\left(a_1\right), f\left(a_2\right)\right) \leq C \cdot d_{\mathcal{A}}\left(a_1, a_2\right)^{\alpha}, \quad \forall a_1, a_2 \in \mathcal{A}.$$

Furthermore, a function is said to be Hölder continuous if it is α-Hölder continuous for some $\alpha \in \mathbb{R}_{>0}$.

- *Lipschitz continuous if it is 1-Hölder continuous, i.e.:*

$$d_{\mathcal{B}}\left(f\left(a_1\right), f\left(a_2\right)\right) \leq C \cdot d_{\mathcal{A}}\left(a_1, a_2\right), \quad \forall a_1, a_2 \in \mathcal{A}.$$

The smallest C for which the above inequality holds is called the Lipschitz constant of $f(\cdot)$.

- *A contraction if there exists a $C \in (0, 1)$ such that:*

$$d_{\mathcal{B}}\left(f\left(a_1\right), f\left(a_2\right)\right) \leq C \cdot d_{\mathcal{A}}\left(a_1, a_2\right), \quad \forall a_1, a_2 \in \mathcal{A}.$$

- *Nonexpansive if:*

$$d_{\mathcal{B}}\left(f\left(a_1\right), f\left(a_2\right)\right) \leq d_{\mathcal{A}}\left(a_1, a_2\right), \quad \forall a_1, a_2 \in \mathcal{A}.$$

As previously pointed out, continuous functions and compact metric spaces play a key role in optimization. First of all, the ranges of continuous functions defined over compact domains are also compact, as claimed by the next Proposition.

Proposition A.6. *Let $\mathcal{A}$ and $\mathcal{B}$ be two metric spaces, and $f : \mathcal{A} \to \mathcal{B}$ be a continuous function. Moreover, let C be a compact subset of $\mathcal{A}$. Then, $f[C]$ is a compact subset of $\mathcal{B}$.*

Now, we can state the Extreme Value Theorem, which gives us sufficient conditions that ensure the existence of a solution for any optimization problem. In particular, here we report its general formulation.

Theorem A.4: Extreme Value Theorem

Let $\mathcal{A}$ be a compact metric space and $f : \mathcal{A} \to \mathbb{R}$ be a continuous function. Then, there exist $a_1, a_2 \in \mathcal{A}$ such that $f(a_1) = \sup f[\mathcal{A}]$ and $f(a_2) = \inf f[\mathcal{A}]$.

Theorem A.4 says that any continuous function whose codomain is the real field $\mathbb{R}$ admits a maximum and a minimum over any compact subset of a metric space. To see this, note that, from Proposition

A.6, $f[\mathcal{A}]$ is a compact subset of $\mathbb{R}$. Therefore, using Proposition A.4, we can deduce that $f[\mathcal{A}]$ is also closed and bounded. Recall that metric-boundedness and order-boundedness coincide for subspaces of $\mathbb{R}$ (see Proposition A.2), therefore $f[\mathcal{A}]$ is order bounded. Finally, due to Remark A.2, we can conclude that $f[\mathcal{A}]$ contains a maximum and minimum, and these are respectively equal to the supremum $\sup f[\mathcal{A}]$ and infimum $\inf f[\mathcal{A}]$.

In practice, continuity is only a sufficient condition for a function $f : \mathcal{A} \to \mathbb{R}$ to assume its maximum (or minimum) over a compact set. A less restrictive form of continuity is the following.

Definition A.22: Semi-continuity of a function. *Let $(\mathcal{A}, d_{\mathcal{A}})$ be any metric space and $f : \mathcal{A} \to \mathbb{R}$. We say that $f(\cdot)$ is upper semi-continuous at $a_1 \in \mathcal{A}$ if, for any $\epsilon \in \mathbb{R}_{>0}$, there exists a $\delta \in \mathbb{R}_{>0}$ (which may depend on both ϵ and a_1) such that:*

$$d_{\mathcal{A}}(a_1, a_2) < \delta \Rightarrow f(a_2) \leq f(a_1) + \epsilon$$

for each $a_2 \in \mathcal{A}$. Similarly, if, for any $\epsilon \in \mathbb{R}_{>0}$, there exists a $\delta \in \mathbb{R}_{>0}$ such that:

$$d_{\mathcal{A}}(a_1, a_2) < \delta \Rightarrow f(a_2) \geq f(a_1) - \epsilon,$$

then $f(\cdot)$ is said to be lower semi-continuous at a_1.
The function $f(\cdot)$ is said to be upper (lower) semi-continuous if it is upper (lower) semi-continuous at each $a_1 \in \mathcal{A}$.

Intuitively speaking, if $f(\cdot)$ is upper semi-continuous at a_1, then the images of the points "nearby" a_1 under $f(\cdot)$ do not exceed $f(a_1)$ by "too much", while there is no restriction about how "far" these images can fall below $f(a_1)$. Vice-versa, if $f(\cdot)$ is lower semi-continuous. The next Remark links Definition A.19 with Definition A.22.

Remark A.10. *Consider a metric space $\mathcal{A}$ and a function $f : \mathcal{A} \to \mathbb{R}$. Then, $f(\cdot)$ is continuous if and only if it is both upper and lower semi-continuous.*

Having defined the semi-continuity of a function, we can generalize the Extreme Value Theorem A.4 as follows.

Theorem A.5: Baire's Theorem on semi-continuous functions

Let $\mathcal{A}$ be a compact metric space and $f : \mathcal{A} \to \mathbb{R}$. We have that:

- *If $f(\cdot)$ is upper semi-continuous, then there exists an $a \in \mathcal{A}$ with $f(a) = \sup f[\mathcal{A}]$,*
- *If $f(\cdot)$ is lower semi-continuous, then there exists an $a \in \mathcal{A}$ with $f(a) = \inf f[\mathcal{A}]$.*

In other words, an upper semi-continuous function always admits a maximum (but not necessarily a minimum) over a compact set. Vice-versa, if $f(\cdot)$ is lower semi-continuous.

Lastly, we conclude our dissertation on continuous functions by stating Tietze Extension Theorem, which is needed for the definition of the preference-based optimization problem.

Theorem A.6: Tietze Extension Theorem

Let $\mathcal{A}$ be a metric space, $\mathcal{B}$ be a closed subset of $\mathcal{A}$ and

$$f : \mathcal{B} \to \mathbb{R}$$

be a continuous function. Then, there exists a continuous extension of $f(\cdot)$ to $\mathcal{A}$, i.e. there exists a function

$$\tilde{f} : \mathcal{A} \to \mathbb{R}$$

that is continuous on $\mathcal{A}$ with $\tilde{f}(b) = f(b), \forall b \in \mathcal{B}$.

A.3 Differentiability of multivariable functions

This book is devoted to solving optimization problems whose objective and constraints functions are multivariable functions, such as $f : \mathbb{R}^n \to \mathbb{R}$. For this reason, we devote this Appendix to review some important concepts related to multivariable functions defined over the n-dimensional Euclidean space. In practice, $\mathbb{R}^n$ is a metric space with distance $d_2(\boldsymbol{x}_1, \boldsymbol{x}_2) = \|\boldsymbol{x}_1 - \boldsymbol{x}_2\|_2$. Therefore, in what follows, we will (implicitly) use many of the notions seen in Appendix A.2.

We start by reviewing the concept of continuity of a function (Definition A.19) under a different light, using the notion of limit. The limit of a multivariable function can be defined analogously to the limit of a sequence (in Appendix A.2.2, Definition A.11).

Definition A.23: Limit of a multivariable function and continuity. *Let $f : \mathcal{A} \to \mathbb{R}, \mathcal{A} \subseteq \mathbb{R}^n$, be a multivariable function and $\langle \boldsymbol{x}_i \rangle_{i\geq 1}$ be a sequence in $\mathcal{A}$. $f(\boldsymbol{x})$ is said to be continuous at $\tilde{\boldsymbol{x}} \in \mathcal{A}$ if, whenever $\langle \boldsymbol{x}_i \rangle_{i\geq 1}$ converges to $\tilde{\boldsymbol{x}}$, the sequence $\langle f(\boldsymbol{x}_i) \rangle_{i\geq 1}$ converges to $\tilde{f} = f(\tilde{\boldsymbol{x}}) \in \mathbb{R}$. In that case, we say that $\tilde{f}$ is the limit of $f(\boldsymbol{x})$ as $\boldsymbol{x} \to \tilde{\boldsymbol{x}}$ and denote it as:*

$$\begin{aligned}\lim_{\boldsymbol{x}\to\tilde{\boldsymbol{x}}} f(\boldsymbol{x}) &= \lim_{i\to\infty} f(\boldsymbol{x}_i)\\ &= f\left(\lim_{i\to\infty} \boldsymbol{x}_i\right)\\ &= \tilde{f}.\end{aligned}$$

The function $f(\cdot)$ is said to be continuous (or continuous everywhere) if it is continuous at each $\tilde{\boldsymbol{x}} \in \mathcal{A}$.

Limits also play a key role in defining the derivatives of a function. Suppose $f : \mathcal{A} \to \mathbb{R}, \mathcal{A} \subseteq \mathbb{R}^n$, then we can compute its partial derivatives at each point in the interior of $\mathcal{A}$ (see Definition A.14).

Definition A.24: Partial derivatives at a point. *Let $f : \mathcal{A} \to \mathbb{R}, \mathcal{A} \subseteq \mathbb{R}^n$, be a multivariable function, $\tilde{\boldsymbol{x}}$ be an interior point of $\mathcal{A}$, i.e. $\tilde{\boldsymbol{x}} \in int_{\mathcal{A}}(\mathcal{A})$, and $\boldsymbol{e}_j$ be the j-th column of the n-dimensional identity matrix. If the limit*

$$\lim_{t \to 0} \frac{f\left(\tilde{\boldsymbol{x}} + t \cdot \boldsymbol{e}_j\right) - f(\tilde{\boldsymbol{x}})}{t} \tag{A.12}$$

exists, then it is called the partial derivative of $f(\boldsymbol{x})$ with respect to $x^{(j)}$ at $\tilde{\boldsymbol{x}}$. We denote it as:

$$\frac{\partial}{\partial x^{(j)}} f(\tilde{\boldsymbol{x}}).$$

In the previous Definition, $\tilde{\boldsymbol{x}} \in \text{int}_{\mathcal{A}}(\mathcal{A})$ to ensure that $\tilde{\boldsymbol{x}} + t \cdot \boldsymbol{e}_j$ in (A.12) lies inside $\mathcal{A}$ for $t \to 0^+$ and $t \to 0^-$, otherwise $f\left(\tilde{\boldsymbol{x}} + t \cdot \boldsymbol{e}_j\right)$ is not even defined.

If $\mathcal{A}$ is an open set (i.e. $\text{int}_{\mathcal{A}}(\mathcal{A}) = \mathcal{A}$) and all the partial derivates of $f(\boldsymbol{x})$ exist at each $\tilde{\boldsymbol{x}} \in \mathcal{A}$, then $\frac{\partial}{\partial x^{(j)}} f(\cdot), j = 1, \ldots, n$, can be seen as multivariable functions in their own right:

$$\frac{\partial}{\partial x^{(j)}} f : \mathcal{A} \to \mathbb{R}.$$

To define the concept of differentiability for multivariable functions, we first need to introduce the notion of linearization of a multivariable function. Consider a function $f : \mathcal{A} \to \mathbb{R}, \mathcal{A} \subseteq \mathbb{R}^n$; we define a linear approximation (or first-order approximation) of $f(\boldsymbol{x})$ near $\tilde{\boldsymbol{x}} \in \mathcal{A}$ as:

$$L'(\boldsymbol{x}) = f(\tilde{\boldsymbol{x}}) + \boldsymbol{d}_f(\tilde{\boldsymbol{x}})^\top \cdot (\boldsymbol{x} - \tilde{\boldsymbol{x}}), \tag{A.13}$$

where $\boldsymbol{d}_f(\tilde{\boldsymbol{x}}) \in \mathbb{R}^n$ is an arbitrary vector which may depend on $\tilde{\boldsymbol{x}}$.

Definition A.25: Differentiability of a multivariable function. *A multivariable function $f : \mathcal{A} \to \mathbb{R}$ on an open set $\mathcal{A} \subseteq \mathbb{R}^n$ is said to be differentiable at $\tilde{\boldsymbol{x}} \in \mathcal{A}$ if there exists a linear approximation $L'(\boldsymbol{x})$ in* (A.13) *for which*

$$\lim_{\boldsymbol{x} \to \tilde{\boldsymbol{x}}} \frac{|f(\boldsymbol{x}) - L'(\boldsymbol{x})|}{\|\boldsymbol{x} - \tilde{\boldsymbol{x}}\|_2} = 0 \tag{A.14}$$

holds.

If $f(\cdot)$ is differentiable at each $\tilde{\boldsymbol{x}} \in \mathcal{A}$, then $f(\cdot)$ is said to be differentiable (or differentiable everywhere).

Remark A.11. *A linear approximation $L'(\boldsymbol{x})$ in* (A.13) *of a multivariable function $f(\boldsymbol{x})$ near a point $\tilde{\boldsymbol{x}}$ that satisfies* (A.14) *is unique, if it exists.*

When (A.14) holds, we refer to $\boldsymbol{d}_f(\tilde{\boldsymbol{x}})$ in (A.13) as the gradient (or first derivative) of $f(\cdot)$ at $\tilde{\boldsymbol{x}}$, denoted as $\nabla_{\boldsymbol{x}} f(\tilde{\boldsymbol{x}})$ and defined as follows:

$$\nabla_{\boldsymbol{x}} f(\tilde{\boldsymbol{x}}) = \begin{bmatrix} \frac{\partial}{\partial x^{(1)}} f(\tilde{\boldsymbol{x}}) \\ \vdots \\ \frac{\partial}{\partial x^{(n)}} f(\tilde{\boldsymbol{x}}) \end{bmatrix}. \tag{A.15}$$

In the multivariable case, the relationship between continuity, differentiability and existence of partial derivatives is more subtle than in the single variable case.

Proposition A.7. *If $f : \mathcal{A} \to \mathbb{R}, \mathcal{A} \subseteq \mathbb{R}^n$, is differentiable at $\tilde{\boldsymbol{x}} \in \mathcal{A}$, then it is continuous at $\tilde{\boldsymbol{x}}$ and its partial derivatives exist at such point.*

Next, we present some important remarks that connect the existence of the partial derivatives of $f(\boldsymbol{x})$ to its differentiability.

Remark A.12. *Consider a function $f : \mathcal{A} \to \mathbb{R}, \mathcal{A} \subseteq \mathbb{R}^n$. We remark that:*

- *The existence of the partial derivatives $\frac{\partial}{\partial x^{(j)}} f(\boldsymbol{x}), j = 1, \ldots, n$, at a point $\tilde{\boldsymbol{x}} \in \mathcal{A}$ does not generally imply the continuity of $f(\boldsymbol{x})$ at that point,*
- *Necessary conditions for the differentiability of $f(\boldsymbol{x})$ at a point $\tilde{\boldsymbol{x}} \in \mathcal{A}$ are: (i) existence of its partial derivatives $\frac{\partial}{\partial x^{(j)}} f(\boldsymbol{x}), j = 1, \ldots, n$, at such point and (ii) continuity of $f(\boldsymbol{x})$ at $\tilde{\boldsymbol{x}}$.*

The following Theorem establishes sufficient conditions for the differentiability of a multivariable function.

Theorem A.7: Sufficient conditions for differentiability of a multivariable function

A multivariable function $f : \mathcal{A} \to \mathbb{R}$ on an open set $\mathcal{A} \subseteq \mathbb{R}^n$ is differentiable if its partial derivatives exist and are continuous functions.

The next Remark shows a useful tool to deal with the differentiation of composite functions.

Remark A.13 (Chain rule)**.** *Let $g : \mathcal{A} \to \mathcal{B}, \mathcal{A} \subseteq \mathbb{R}^n, \mathcal{B} \subseteq \mathbb{R}$, be a multivariable function that is differentiable at a point $\tilde{\boldsymbol{x}} \in \mathcal{A}$ and let $h : \mathcal{B} \to \mathbb{R}$ be a single variable function that is differentiable at $\tilde{t} = g(\tilde{\boldsymbol{x}})$. Consider the composite function $f : \mathcal{A} \to \mathbb{R}$, $f(\boldsymbol{x}) = h(g(\boldsymbol{x}))$; we can use the chain rule to compute the gradient of $f(\boldsymbol{x})$ at $\tilde{\boldsymbol{x}}$ as follows:*

$$\nabla_{\boldsymbol{x}} f(\tilde{\boldsymbol{x}}) = \frac{d}{dt} h(\tilde{t}) \bigg|_{\tilde{t} = g(\tilde{\boldsymbol{x}})} \cdot \nabla_{\boldsymbol{x}} g(\tilde{\boldsymbol{x}}).$$

The second derivative of a multivariable function $f : \mathcal{A} \to \mathbb{R}$ on an open set $\mathcal{A} \subseteq \mathbb{R}^n$ can be defined analogously to the gradient. This time, we consider the second-order approximation of $f(\boldsymbol{x})$ near $\tilde{\boldsymbol{x}} \in \mathcal{A}$:

$$L''(\boldsymbol{x}) = f(\tilde{\boldsymbol{x}}) + \boldsymbol{d}_f(\tilde{\boldsymbol{x}})^\top \cdot (\boldsymbol{x} - \tilde{\boldsymbol{x}}) + \frac{1}{2} \cdot (\boldsymbol{x} - \tilde{\boldsymbol{x}})^\top \cdot D_f(\tilde{\boldsymbol{x}}) \cdot (\boldsymbol{x} - \tilde{\boldsymbol{x}}), \tag{A.16}$$

where $D_f(\tilde{\boldsymbol{x}}) \in \mathbb{R}^{n \times n}$ is an arbitrary matrix which may depend on $\tilde{\boldsymbol{x}}$ and $\boldsymbol{d}_f(\tilde{\boldsymbol{x}})$. Then, $f(\boldsymbol{x})$ is twice differentiable at $\tilde{\boldsymbol{x}}$ if there exists a second-order approximation that satisfies:

$$\lim_{\boldsymbol{x} \to \tilde{\boldsymbol{x}}} \frac{|f(\boldsymbol{x}) - L''(\boldsymbol{x})|}{\|\boldsymbol{x} - \tilde{\boldsymbol{x}}\|_2^2} = 0. \tag{A.17}$$

When there exists an $L''(\boldsymbol{x})$ in (A.16) for which (A.17) holds, we refer to $\boldsymbol{d}_f(\tilde{\boldsymbol{x}})$ as the gradient (or first derivative) of $f(\cdot)$ at $\tilde{\boldsymbol{x}}$ (defined as in (A.15)), while $D_f(\tilde{\boldsymbol{x}})$ is the Hessian (or second derivative) of $f(\cdot)$ at $\tilde{\boldsymbol{x}}$. In particular, the Hessian of $f(\cdot)$ at $\tilde{\boldsymbol{x}}$ is defined as follows:

$$\nabla^2_{\boldsymbol{xx}} f(\tilde{\boldsymbol{x}}) = \begin{bmatrix} \frac{\partial^2}{\partial \left(x^{(1)}\right)^2} f(\tilde{\boldsymbol{x}}) & \frac{\partial^2}{\partial x^{(1)} \partial x^{(2)}} f(\tilde{\boldsymbol{x}}) & \cdots & \frac{\partial^2}{\partial x^{(1)} \partial x^{(n)}} f(\tilde{\boldsymbol{x}}) \\ \frac{\partial^2}{\partial x^{(2)} \partial x^{(1)}} f(\tilde{\boldsymbol{x}}) & \frac{\partial^2}{\partial \left(x^{(2)}\right)^2} f(\tilde{\boldsymbol{x}}) & \cdots & \frac{\partial^2}{\partial x^{(2)} \partial x^{(n)}} f(\tilde{\boldsymbol{x}}) \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2}{\partial x^{(n)} \partial x^{(1)}} f(\tilde{\boldsymbol{x}}) & \frac{\partial^2}{\partial x^{(n)} \partial x^{(2)}} f(\tilde{\boldsymbol{x}}) & \cdots & \frac{\partial^2}{\partial \left(x^{(n)}\right)^2} f(\tilde{\boldsymbol{x}}) \end{bmatrix}, \tag{A.18}$$

where $\frac{\partial^2}{\partial x^{(i)} \partial x^{(j)}} f(\tilde{\boldsymbol{x}})$ is the partial derivative of $\frac{\partial}{\partial x^{(i)}} f(\boldsymbol{x})$ with respect to $x^{(j)}$ at the point $\tilde{\boldsymbol{x}}$ (second-order partial derivative). If $f(\cdot)$ is twice differentiable at each $\tilde{\boldsymbol{x}} \in \mathcal{A}$, then $f(\cdot)$ is said to be twice differentiable.

Lastly, the next Definition generalizes the concepts seen in this Appendix to higher-order derivatives.

Definition A.26: Differentiability classes. *Let $k \in \mathbb{N} \cup \{0\}$ and let $f : \mathcal{A} \to \mathbb{R}$ be a multivariable function on an open set $\mathcal{A} \subseteq \mathbb{R}^n$. $f(\cdot)$ is said to be of differentiability class $C^k(\mathcal{A})$ if all its partial derivatives*

$$\frac{\partial^k}{\partial \left(x^{(1)}\right)^{k_1} \partial \left(x^{(2)}\right)^{k_2} \cdots \partial \left(x^{(n)}\right)^{k_n}} f(\tilde{\boldsymbol{x}})$$

exist and are continuous for every $k_1, \ldots, k_n \in \mathbb{N} \cup \{0\}$ such that $k_1 + \ldots + k_n \leq k$, and for each $\tilde{\boldsymbol{x}} \in \mathcal{A}$. We denote it as $f(\boldsymbol{x}) \in C^k(\mathcal{A})$.

Notably:

- *$C^0(\mathcal{A})$ is the class of all continuous functions;*
- *$C^1(\mathcal{A})$ is the class of all continuously differentiable functions;*
- *$C^\infty(\mathcal{A})$ is the class of all infinitely differentiable functions (also called smooth functions or infinitely smooth functions).*

A.4 Optimization concepts

This Appendix is devoted to reviewing some basic optimization concepts. Without loss of generality, we only consider the minimization of some cost function (possibly) over some constraint set. The unconstrained and constrained optimization problems are introduced; their minimizers/solutions are identified. We report first-order and second-order necessary and sufficient conditions that characterize both the local minimizers of unconstrained optimization problems and the local solutions of constrained optimization problems. We also give some remarks on convexity.

Assumption A.1 (Assumptions on the cost function and constraints functions)**.** *In what follows, we assume that the multivariable functions* $f : \mathbb{R}^n \to \mathbb{R}$ *and* $g^{(i)} : \mathbb{R}^n \to \mathbb{R}, i = 1, \ldots, q, q \in \mathbb{N}$, *are twice differentiable.*

A.4.1 Unconstrained optimization

The unconstrained optimization problem is defined as follows:

$$\begin{aligned} &\arg \min_{\boldsymbol{x}} f(\boldsymbol{x}) \\ &\text{s.t.} \quad \boldsymbol{x} \in \mathbb{R}^n, \end{aligned} \tag{A.19}$$

where $\boldsymbol{x} \in \mathbb{R}^n$ is the decision vector and $f : \mathbb{R}^n \to \mathbb{R}$ is the cost function.
A point $\boldsymbol{x}^+ \in \mathbb{R}^n$ is said to be:

- A global minimizer of $f(\boldsymbol{x})$ if $f(\boldsymbol{x}^+) \leq f(\boldsymbol{x}), \forall \boldsymbol{x} \in \mathbb{R}^n$;
- A (weak) local minimizer of $f(\boldsymbol{x})$ if there exists a neighborhood of $\boldsymbol{x}^+$, $\mathcal{N}(\boldsymbol{x}^+)$, such that $f(\boldsymbol{x}^+) \leq f(\boldsymbol{x}), \forall \boldsymbol{x} \in \mathcal{N}(\boldsymbol{x}^+)$;
- A strict local minimizer of $f(\boldsymbol{x})$ if there exists a neighborhood of $\boldsymbol{x}^+$, $\mathcal{N}(\boldsymbol{x}^+)$, such that $f(\boldsymbol{x}^+) < f(\boldsymbol{x}), \forall \boldsymbol{x} \in \mathcal{N}(\boldsymbol{x}^+)$ with $\boldsymbol{x} \neq \boldsymbol{x}^+$;
- An isolated local minimizer of $f(\boldsymbol{x})$ if there exists a neighborhood of $\boldsymbol{x}^+$, $\mathcal{N}(\boldsymbol{x}^+)$, such that $\boldsymbol{x}^+$ is the only local minimizer in $\mathcal{N}(\boldsymbol{x}^+)$.

Clearly, any global minimizer of $f(\boldsymbol{x})$ is also a local minimizer of $f(\boldsymbol{x})$ but the opposite is not necessarily true.
Now, we review first-order and second-order conditions that characterize the local minimizers of Problem (A.19). First-order conditions make use of the gradient of the cost function, $\nabla_{\boldsymbol{x}} f(\boldsymbol{x})$, whereas second-order conditions also consider its Hessian $\nabla^2_{\boldsymbol{xx}} f(\boldsymbol{x})$.

Theorem A.8: First-order necesssary conditions for optimality

If $\boldsymbol{x}^{+}$ is a local minimizer of $f(\boldsymbol{x})$ and $f(\boldsymbol{x})$ is continuously differentiable in an open neighborhood of $\boldsymbol{x}^{+}$, then:

$$\nabla_{\boldsymbol{x}} f\left(\boldsymbol{x}^{+}\right) = \mathbf{0}_n.$$

Theorem A.9: Second-order necesssary conditions for optimality

If $\boldsymbol{x}^{+}$ is a local minimizer of $f(\boldsymbol{x})$ and $\nabla^2_{\boldsymbol{xx}} f(\boldsymbol{x})$ is continuous in an open neighborhood of $\boldsymbol{x}^{+}$, then:

1. $\nabla_{\boldsymbol{x}} f(\boldsymbol{x}^{+}) = \mathbf{0}_n$,
2. $\nabla^2_{\boldsymbol{xx}} f(\boldsymbol{x}^{+})$ *is positive semidefinite.*

Theorem A.10: Second-order sufficient conditions for optimality

Suppose that:

1. $\nabla_{\boldsymbol{x}} f(\boldsymbol{x}^{+}) = \mathbf{0}_n$,
2. $\nabla^2_{\boldsymbol{xx}} f(\boldsymbol{x})$ *is continuous in an open neighborhood of* $\boldsymbol{x}^{+}$,
3. $\nabla^2_{\boldsymbol{xx}} f(\boldsymbol{x}^{+})$ *is positive definite.*

Then, $\boldsymbol{x}^{+}$ is a strict local minimizer of $f(\boldsymbol{x})$.

A.4.2 Constrained optimization

The constrained optimization problem is defined as follows:

$$\begin{aligned} &\arg\min_{\boldsymbol{x}} f(\boldsymbol{x}) && \\ \text{s.t.}\quad & g^{(i)}(\boldsymbol{x}) \leq 0 && i \in \mathcal{I}_{ineq} \\ & g^{(i)}(\boldsymbol{x}) = 0 && i \in \mathcal{I}_{eq}, \end{aligned} \tag{A.20}$$

where $g^{(i)} : \mathbb{R}^n \rightarrow \mathbb{R}$ is the i-th constraint function, $i \in \mathcal{I}_{ineq} \cup \mathcal{I}_{eq} = \{1, \ldots, q\}, q \in \mathbb{N}$. The inequalities $g^{(i)}(\boldsymbol{x}) \leq 0$ and the equalities $g^{(i)}(\boldsymbol{x}) = 0$ constitute the constraints of the optimization problem, of which there are a total of q. The constraints can be grouped inside a set Ω, called the constraint set, defined as:

$$\Omega = \left\{\boldsymbol{x} : g^{(i)}(\boldsymbol{x}) \leq 0, \forall i \in \mathcal{I}_{ineq}, g^{(i)}(\boldsymbol{x}) = 0, \forall i \in \mathcal{I}_{eq}\right\}. \tag{A.21}$$

Then, Problem (A.20) can be re-written as:

$$\begin{aligned} &\arg\min_{\boldsymbol{x}} f(\boldsymbol{x}) \\ &\text{s.t.} \quad \boldsymbol{x} \in \Omega. \end{aligned} \tag{A.22}$$

A point $\boldsymbol{x} \in \mathbb{R}^n$ is said to be feasible if $\boldsymbol{x} \in \Omega$, otherwise it is infeasible. For this reason, Ω is also referred to as the feasible set (or feasible region) of the optimization problem.
The distinction between the types of solutions of Problem (A.22) is analogous to the different types of minimizers for the unconstrained optimization case, reported in Appendix A.4.1. Formally, a point $\boldsymbol{x}^+ \in \mathbb{R}^n$ is said to be:

- A global solution of Problem (A.22) if $f(\boldsymbol{x}^+) \leq f(\boldsymbol{x}), \forall \boldsymbol{x} \in \Omega$;
- A (weak) local solution of Problem (A.22) if $\boldsymbol{x}^+ \in \Omega$ and there exists a neighborhood of $\boldsymbol{x}^+$, $\mathcal{N}(\boldsymbol{x}^+)$, such that $f(\boldsymbol{x}^+) \leq f(\boldsymbol{x}), \forall \boldsymbol{x} \in \mathcal{N}(\boldsymbol{x}^+) \cap \Omega$;
- A strict local solution of Problem (A.22) if $\boldsymbol{x}^+ \in \Omega$ and there exists a neighborhood of $\boldsymbol{x}^+$, $\mathcal{N}(\boldsymbol{x}^+)$, such that $f(\boldsymbol{x}^+) < f(\boldsymbol{x}), \forall \boldsymbol{x} \in \mathcal{N}(\boldsymbol{x}^+) \cap \Omega$ with $\boldsymbol{x} \neq \boldsymbol{x}^+$;
- An isolated local solution of Problem (A.22) if $\boldsymbol{x}^+ \in \Omega$ and there exists a neighborhood of $\boldsymbol{x}^+$, $\mathcal{N}(\boldsymbol{x}^+)$, such that $\boldsymbol{x}^+$ is the only local minimizer in $\mathcal{N}(\boldsymbol{x}^+) \cap \Omega$.

Notation and conventions. Throughout this book, we will use the terms "minimizer" and "solution" interchangeably. In particular, to be precise:

- A minimizer of $f(\boldsymbol{x})$ is a solution of the unconstrained problem in (A.19),
- A minimizer of Problem (A.20) is a solution of the constrained optimization problem.

Furthermore, notation-wise, we will denote a global minimizer of an optimization problem as $\boldsymbol{x}^*$ (or $\boldsymbol{x}_i^*$) to distinguish it from a local optimizer $\boldsymbol{x}^+$.
In order to state necessary and sufficient conditions that characterize the local solutions of Problem (A.20), we need to review several different concepts. First of all, we must distinguish between active and inactive inequality constraints. Secondly, suitable "regularity" conditions on the constraints (often referred to as constraint qualification conditions) are needed [50]. Lastly, we have to introduce the Lagrangian function, which plays a key role in constrained optimization.

Definition A.27: Active set. *At a feasible point $\tilde{x} \in \Omega$, the i-th inequality constraint, i.e. $g^{(i)}(\cdot), i \in \mathcal{I}_{ineq}$, is said to be:*

- *Active if it holds with equality, i.e.* $g^{(i)}(\tilde{\boldsymbol{x}}) = 0$*,*
- *Inactive if the strict inequality* $g^{(i)}(\tilde{\boldsymbol{x}}) < 0$ *is satisfied.*

The active set $\mathcal{A}(\tilde{\boldsymbol{x}}) \subseteq \{1, \ldots, q\}$ *at any feasible point* $\tilde{\boldsymbol{x}} \in \Omega$ *is defined as:*

$$\mathcal{A}(\tilde{\boldsymbol{x}}) = \mathcal{I}_{eq} \cup \left\{ i : g^{(i)}(\tilde{\boldsymbol{x}}) = 0, i \in \mathcal{I}_{ineq} \right\}. \tag{A.23}$$

In practice, given a feasible point $\tilde{\boldsymbol{x}} \in \Omega$, $\mathcal{A}(\tilde{\boldsymbol{x}})$ contains the indices of those constraints which hold with equality at $\tilde{\boldsymbol{x}}$.

Definition A.28: LICQ. *Given a feasible point* $\tilde{\boldsymbol{x}} \in \Omega$ *and the active set* $\mathcal{A}(\tilde{\boldsymbol{x}})$ *in* (A.23)*, we say that the Linear Independence Constraint Qualification (LICQ) condition holds at* $\tilde{\boldsymbol{x}}$ *if the gradients of the active constraints functions, namely* $\nabla_{\boldsymbol{x}} g^{(i)}(\cdot), i \in \mathcal{A}(\tilde{\boldsymbol{x}})$*, are linearly independent at* $\tilde{\boldsymbol{x}}$*.*

Definition A.29: Lagrangian function. *The Lagrangian function is defined as:*

$$\mathcal{L}(\boldsymbol{x}, \boldsymbol{\lambda}) = f(\boldsymbol{x}) + \sum_{i=1}^{q} \lambda^{(i)} \cdot g^{(i)}(\boldsymbol{x}), \tag{A.24}$$

where $\boldsymbol{\lambda} \in \mathbb{R}^q$ *is the vector of Lagrange multipliers (one for each constraint).*

We can finally state the widely known Karush-Kuhn-Tucker (KKT) conditions, which are the first-order necessary conditions that characterize the local solutions of Problem (A.20).

Theorem A.11: First-order necessary conditions for optimality (KKT conditions)

Suppose that $\boldsymbol{x}^+$ *is a local solution of Problem* (A.20) *and that the LICQ condition holds at* $\boldsymbol{x}^+$*. Then,* $\exists \boldsymbol{\lambda}^+ \in \mathbb{R}^q, \boldsymbol{\lambda}^+ = \begin{bmatrix} \lambda^{+(1)} & \ldots & \lambda^{+(q)} \end{bmatrix}^\top$*, such that:*

$$\nabla_{\boldsymbol{x}} \mathcal{L}\left(\boldsymbol{x}^+, \boldsymbol{\lambda}^+\right) = \boldsymbol{0}_n \tag{A.25a}$$

$$g^{(i)}\left(\boldsymbol{x}^+\right) \leq 0 \qquad \forall i \in \mathcal{I}_{ineq} \tag{A.25b}$$

$$g^{(i)}\left(\boldsymbol{x}^+\right) = 0 \qquad \forall i \in \mathcal{I}_{eq} \tag{A.25c}$$

$$\lambda^{+(i)} \geq 0 \qquad \forall i \in \mathcal{I}_{ineq} \tag{A.25d}$$

$$\lambda^{+(i)} \cdot g^{(i)}\left(\boldsymbol{x}^+\right) = 0 \qquad \forall i \in \mathcal{I}_{ineq} \cup \mathcal{I}_{eq}. \tag{A.25e}$$

In particular, the condition in (A.25e) *is referred to as complementary slackness condition.*

Remark A.14. *From* (A.25e)*, the Lagrange multipliers $\lambda^{+^{(i)}}$ associated to the inactive constraints at $\boldsymbol{x}^+$, i.e. $g^{(i)}\left(\boldsymbol{x}^+\right)$ such that $i \notin \mathcal{A}\left(\boldsymbol{x}^+\right)$, are zero. Then, the condition in* (A.25a) *becomes:*

$$\begin{aligned}\nabla_{\boldsymbol{x}}\mathcal{L}\left(\boldsymbol{x}^+, \boldsymbol{\lambda}^+\right) &= \mathbf{0}_n \\ \nabla_{\boldsymbol{x}} f\left(\boldsymbol{x}^+\right) + \sum_{i=1}^{q} \lambda^{+^{(i)}} \cdot \nabla_{\boldsymbol{x}} g^{(i)}\left(\boldsymbol{x}^+\right) &= \mathbf{0}_n \\ \nabla_{\boldsymbol{x}} f\left(\boldsymbol{x}^+\right) + \sum_{i \in \mathcal{A}\left(\boldsymbol{x}^+\right)} \lambda^{+^{(i)}} \cdot \nabla_{\boldsymbol{x}} g^{(i)}\left(\boldsymbol{x}^+\right) &= \mathbf{0}_n.\end{aligned}$$

Thus, at a local solution $\boldsymbol{x}^+ \in \Omega$ of Problem (A.20) *at which the LICQ condition holds, we have:*

$$\nabla_{\boldsymbol{x}} f\left(\boldsymbol{x}^+\right) = -\sum_{i \in \mathcal{A}\left(\boldsymbol{x}^+\right)} \lambda^{+^{(i)}} \cdot \nabla_{\boldsymbol{x}} g^{(i)}\left(\boldsymbol{x}^+\right). \tag{A.26}$$

Remark A.15. *When the LICQ condition holds at $\boldsymbol{x}^+$, $\boldsymbol{\lambda}^+$ is unique.*

Before stating the second-order conditions, further notions are required. First of all, we must define the tangent cone to the feasible set at a point.

Definition A.30: Tangent cone to the feasible set at a point. *Consider a point $\tilde{\boldsymbol{x}} \in \mathbb{R}^n$ at which the LICQ condition holds and with corresponding active set $\mathcal{A}\left(\tilde{\boldsymbol{x}}\right)$. The tangent cone to the feasible set Ω in* (A.21) *at $\tilde{\boldsymbol{x}}$ is defined as:*

$$\begin{aligned}\mathcal{F}_1\left(\tilde{\boldsymbol{x}}\right) = \Big\{\alpha \cdot \boldsymbol{d} : \alpha \in \mathbb{R}_{>0}, \boldsymbol{d} \in \mathbb{R}^n, \\ \boldsymbol{d}^\top \cdot \nabla_{\boldsymbol{x}} g^{(i)}\left(\tilde{\boldsymbol{x}}\right) = 0, \forall i \in \mathcal{I}_{eq}, \\ \boldsymbol{d}^\top \cdot \nabla_{\boldsymbol{x}} g^{(i)}\left(\tilde{\boldsymbol{x}}\right) \leq 0, \forall i \in \mathcal{A}\left(\tilde{\boldsymbol{x}}\right) \cap \mathcal{I}_{ineq}\Big\}.\end{aligned} \tag{A.27}$$

In practice, $\mathcal{F}_1\left(\tilde{\boldsymbol{x}}\right)$ in (A.27) is a set which contains all *feasible directions* of Ω at $\tilde{\boldsymbol{x}}$.

Let $\boldsymbol{x}^+, \boldsymbol{\lambda}^+$ satisfy the KKT conditions in (A.25) and $\boldsymbol{d} \in \mathcal{F}_1\left(\boldsymbol{x}^+\right)$. We multiply both sides of (A.26) by $\boldsymbol{d}^\top$, obtaining:

$$\begin{aligned}\boldsymbol{d}^\top \cdot \nabla_{\boldsymbol{x}} f\left(\boldsymbol{x}^+\right) &= -\sum_{i \in \mathcal{A}\left(\boldsymbol{x}^+\right)} \lambda^{+^{(i)}} \cdot \boldsymbol{d}^\top \cdot \nabla_{\boldsymbol{x}} g^{(i)}\left(\boldsymbol{x}^+\right) \\ &= -\sum_{i \in \mathcal{I}_{eq}} \lambda^{+^{(i)}} \cdot \underbrace{\boldsymbol{d}^\top \cdot \nabla_{\boldsymbol{x}} g^{(i)}\left(\boldsymbol{x}^+\right)}_{=0} - \sum_{i \in \mathcal{A}\left(\boldsymbol{x}^+\right) \cap \mathcal{I}_{ineq}} \underbrace{\lambda^{+^{(i)}}}_{\geq 0} \cdot \underbrace{\boldsymbol{d}^\top \cdot \nabla_{\boldsymbol{x}} g^{(i)}\left(\boldsymbol{x}^+\right)}_{\leq 0} \\ &= -\sum_{i \in \mathcal{A}\left(\boldsymbol{x}^+\right) \cap \mathcal{I}_{ineq}} \underbrace{\lambda^{+^{(i)}}}_{\geq 0} \cdot \underbrace{\boldsymbol{d}^\top \cdot \nabla_{\boldsymbol{x}} g^{(i)}\left(\boldsymbol{x}^+\right)}_{\leq 0} \\ &\geq 0.\end{aligned}$$

Therefore, two situations can arise: either $\boldsymbol{d}^\top \cdot \nabla_{\boldsymbol{x}} f\left(\boldsymbol{x}^+\right) > 0$ or $\boldsymbol{d}^\top \cdot \nabla_{\boldsymbol{x}} f\left(\boldsymbol{x}^+\right) = 0$. In the former case, there is no move along $\boldsymbol{d}$ that will decrease $f\left(\boldsymbol{x}\right)$. Instead, in the latter case, we cannot determine whether a move along $\boldsymbol{d}$ will increase or decrease $f(\boldsymbol{x})$ using only first derivative information. We group all the directions for which $\boldsymbol{d}^\top \cdot \nabla_{\boldsymbol{x}} f(\boldsymbol{x}^+) = 0$ holds inside the critical cone.

Definition A.31: Critical cone. *Let $\boldsymbol{x}^+, \boldsymbol{\lambda}^+$ satisfy the KKT conditions in* (A.25). *The critical cone $\mathcal{F}_2(\boldsymbol{x}^+, \boldsymbol{\lambda}^+)$ is defined as:*

$$\begin{aligned}\mathcal{F}_2(\boldsymbol{x}^+, \boldsymbol{\lambda}^+) = \Big\{\boldsymbol{d} : \boldsymbol{d} \in \mathcal{F}_1(\boldsymbol{x}^+), \boldsymbol{d}^\top \cdot \nabla_{\boldsymbol{x}} g^{(i)}(\boldsymbol{x}^+) = 0,& \\ \forall i \in \mathcal{A}(\boldsymbol{x}^+) \cap \mathcal{I}_{ineq} \text{ with } \lambda^{+(i)} > 0\Big\},&\end{aligned}$$

or equivalently:

$$\begin{aligned}\mathcal{F}_2(\boldsymbol{x}^+, \boldsymbol{\lambda}^+) = \Big\{\alpha \cdot \boldsymbol{d} : \alpha \in \mathbb{R}_{>0}, \boldsymbol{d} \in \mathbb{R}^n,& \\ \boldsymbol{d}^\top \cdot \nabla_{\boldsymbol{x}} g^{(i)}(\boldsymbol{x}^+) = 0, \forall i \in \mathcal{I}_{eq},& \\ \boldsymbol{d}^\top \cdot \nabla_{\boldsymbol{x}} g^{(i)}(\boldsymbol{x}^+) = 0, \forall i \in \mathcal{A}(\boldsymbol{x}^+) \cap \mathcal{I}_{ineq} \text{ with } \lambda^{+(i)} > 0,& \\ \boldsymbol{d}^\top \cdot \nabla_{\boldsymbol{x}} g^{(i)}(\boldsymbol{x}^+) \le 0, \forall i \in \mathcal{A}(\boldsymbol{x}^+) \cap \mathcal{I}_{ineq} \text{ with } \lambda^{+(i)} = 0\Big\}.&\end{aligned} \tag{A.28}$$

Having defined the critical cone, we can finally state the second-order necessary and sufficient conditions which characterize the local solutions of Problem (A.20).

Theorem A.12: Second-order necessary conditions for optimality

Suppose that $\boldsymbol{x}^+$ is a local solution of Problem (A.20) *and let the LICQ condition hold at $\boldsymbol{x}^+$. Moreover, let $\boldsymbol{\lambda}^+$ be the vector of Lagrange multipliers that satisfies the KKT conditions in* (A.25). *Then:*

$$\boldsymbol{d}^\top \cdot \nabla^2_{\boldsymbol{xx}} \mathcal{L}(\boldsymbol{x}^+, \boldsymbol{\lambda}^+) \cdot \boldsymbol{d} \ge 0, \quad \forall \boldsymbol{d} \in \mathcal{F}_2(\boldsymbol{x}^+, \boldsymbol{\lambda}^+).$$

Theorem A.13: Second-order sufficient conditions for optimality

Suppose that, for some feasible point $\boldsymbol{x}^+ \in \Omega$, there exists a vector of Lagrange multipliers $\boldsymbol{\lambda}^+$ that satisfies the KKT conditions in (A.25). *Suppose also that:*

$$\boldsymbol{d}^\top \cdot \nabla^2_{\boldsymbol{xx}} \mathcal{L}(\boldsymbol{x}^+, \boldsymbol{\lambda}^+) \cdot \boldsymbol{d} > 0, \quad \forall \boldsymbol{d} \in \mathcal{F}_2(\boldsymbol{x}^+, \boldsymbol{\lambda}^+), \boldsymbol{d} \ne \boldsymbol{0}_n.$$

Then $\boldsymbol{x}^+$ is a strict local solution of Problem (A.20).

A.4.3 Convex optimization problems

Convexity is a desirable property for any optimization problem. There exist very efficient algorithms, such as interior-point methods, that accurately solve convex optimization problems in polynomial time.

To define convex optimization problems, we require the notion of convexity for sets and functions.

Definition A.32: Convex set. *A set $\mathcal{A} \subseteq \mathbb{R}^n$ is said to be convex if, for any $\boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathcal{A}$ and any $\theta \in [0, 1]$, we have:*

$$\theta \cdot \boldsymbol{x}_1 + (1 - \theta) \cdot \boldsymbol{x}_2 \in \mathcal{A}.$$

Alternatively, we could say that a subset $\mathcal{A}$ of an Euclidean space is convex if the line segment that connects any two points in $\mathcal{A}$ lies in $\mathcal{A}$. The next Example shows some commonly used convex sets.

Example A.6: Important convex sets

The following sets are convex:

- The empty set $\emptyset$;
- Any Euclidean space $\mathbb{R}^n$;
- Any hyperplane $\mathcal{A} = \{\boldsymbol{x} : \boldsymbol{a}^\top \cdot \boldsymbol{x} = \boldsymbol{b}, \boldsymbol{x} \in \mathbb{R}^n\}$, where $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{R}^n$;
- Any open and closed ball in $\mathbb{R}^n$, namely $\mathcal{B}(\tilde{\boldsymbol{x}}; \epsilon)$ in (A.10) and $\bar{\mathcal{B}}(\tilde{\boldsymbol{x}}; \epsilon)$ in (A.11).

Next, we define what is means for a function to be convex.

Definition A.33: Convex and concave functions. *A function $f : \mathcal{A} \to \mathbb{R}, \mathcal{A} \subseteq \mathbb{R}^n$, is convex if $\mathcal{A}$ is a convex set and if, for all $\boldsymbol{x}_i, \boldsymbol{x}_j \in \mathcal{A}$ and $\theta \in [0, 1]$, we have:*

$$f\left(\theta \cdot \boldsymbol{x}_i + (1 - \theta) \cdot \boldsymbol{x}_j\right) \leq \theta \cdot f(\boldsymbol{x}_i) + (1 - \theta) \cdot f\left(\boldsymbol{x}_j\right).$$

If the previous inequality holds whenever $\boldsymbol{x}_i \neq \boldsymbol{x}_j$ and $\theta \in (0, 1)$, then $f(\boldsymbol{x})$ is said to be strictly convex.

We say that $f(\boldsymbol{x})$ is (strictly) concave if $-f(\boldsymbol{x})$ is (strictly) convex.

Geometrically, a convex function $f(\boldsymbol{x})$ is such that the line segment between $(\boldsymbol{x}_i, f(\boldsymbol{x}_i))$ and $\left(\boldsymbol{x}_j, f\left(\boldsymbol{x}_j\right)\right)$ lies above the graph of $f(\boldsymbol{x})$. There exist first-order and second-order conditions that let us assess the convexity of a function.

Theorem A.14: First-order condition for convexity

Let $f : \mathcal{A} \to \mathbb{R}, \mathcal{A} \subseteq \mathbb{R}^n$, be a differentiable function. Then, $f(\boldsymbol{x})$ is convex if and only if $\mathcal{A}$ is a convex set and

$$f\left(\boldsymbol{x}_j\right) \geq f(\boldsymbol{x}_i) + \nabla_{\boldsymbol{x}} f(\boldsymbol{x}_i)^\top \cdot \left(\boldsymbol{x}_j - \boldsymbol{x}_i\right)$$

holds for all $\boldsymbol{x}_i, \boldsymbol{x}_j \in \mathcal{A}$.

Theorem A.15: Second-order condition for convexity

Let $f : \mathcal{A} \rightarrow \mathbb{R}, \mathcal{A} \subseteq \mathbb{R}^n$, be a twice differentiable function. Then, $f(\boldsymbol{x})$ is convex if and only if $\mathcal{A}$ is a convex set and its Hessian $\nabla^2_{\boldsymbol{xx}} f(\boldsymbol{x})$ is positive semidefinite $\forall \boldsymbol{x} \in \mathcal{A}$.

We are now ready to define convex optimization problems.

Definition A.34: Convex optimization problem. *An optimization problem, either unconstrained as in Problem* (A.19) *or constrained as in Problem* (A.20)*, is convex if and only if:*

- *The cost function $f(\boldsymbol{x})$ is convex;*
- *The inequality constraints functions $g^{(i)}(\boldsymbol{x}), i \in \mathcal{I}_{ineq}$, are convex;*
- *The equality constraints functions $g^{(i)}(\boldsymbol{x}), i \in \mathcal{I}_{eq}$, are affine, i.e. $g^{(i)}(\boldsymbol{x}) = \boldsymbol{a}_i^\top \cdot \boldsymbol{x} - \boldsymbol{b}_i$ with $\boldsymbol{a}_i, \boldsymbol{b}_i \in \mathbb{R}^n$.*

In particular, the last two conditions make the constraint set Ω in (A.21) *convex.*

The main advantage of convex optimization problems over non-convex ones is highlighted by the following Proposition.

Proposition A.8: Global minimizers of convex optimization problems. *If Problem* (A.19) *(or Problem* (A.20)*) is convex, then any locally optimal solution is also globally optimal.*

In practice, this means that if we know a-priori that we are dealing with a convex optimization problem, it does not make sense to use a global optimization procedure to solve it. Instead, we should employ an efficient local optimization procedure which, in this case, is guaranteed to find the global minima.

Appendix B. Benchmark optimization problems

This Appendix reports the benchmark global optimization problems used to compare the performances of the surrogate-based methods proposed in this book against the original algorithms (see Chapter 7).

B.1 Unconstrained optimization problems

1. `bemporad` [10]:

   dimensionality: $n = 1$

   cost function: $f(x) = \left[1 + \frac{x^{(1)} \cdot \sin\left(2 \cdot x^{(1)}\right) \cdot \cos\left(3 \cdot x^{(1)}\right)}{1 + \left(x^{(1)}\right)^2}\right]^2 + \frac{\left(x^{(1)}\right)^2}{12} + \frac{x^{(1)}}{10}$

   constraints: $\Omega = \{x : l \leq x \leq u\}$, $\Xi = \mathbb{R}^n$

   ▹ $l = -3, u = 3$

   minimizer: $x^* = -0.9599$

   minimum: $f^* = 0.2795$

2. `gramacy and lee` [53]:

   dimensionality: $n = 1$

   cost function: $f(x) = \frac{\sin\left(10 \cdot \pi \cdot x^{(1)}\right)}{2 \cdot x^{(1)}} + \left(x^{(1)} - 1\right)^4$

   constraints: $\Omega = \{x : l \leq x \leq u\}$, $\Xi = \mathbb{R}^n$

   ▹ $l = 0.5, u = 2.5$

   minimizer: $x^* = 0.5486$

   minimum: $f^* = -0.8690$

3. `ackley` [62]:

   dimensionality: $n = 2$

   cost function: $f(\boldsymbol{x}) = -20 \cdot \exp\left\{-0.02 \cdot \sqrt{n^{-1} \cdot \sum_{i=1}^{n} \left(x^{(i)}\right)^2}\right\} + - \exp\left\{n^{-1} \cdot \sum_{i=1}^{n} \cos\left(2 \cdot \pi \cdot x^{(i)}\right)\right\} + 20 + \exp\{1\}$

   constraints: $\Omega = \{\boldsymbol{x} : \boldsymbol{l} \leq \boldsymbol{x} \leq \boldsymbol{u}\}$, $\Xi = \mathbb{R}^n$

   ▹ $\boldsymbol{l} = -35 \cdot \mathbf{1}_n, \boldsymbol{u} = 35 \cdot \mathbf{1}_n$

   minimizer: $\boldsymbol{x}^* = \mathbf{0}_n$

minimum: $f^* = 0$

4. `bukin 6` [62]:

   dimensionality: $n = 2$

   cost function: $f(\boldsymbol{x}) = 100 \cdot \sqrt{\left\| x^{(2)} - 0.01 \cdot \left(x^{(1)}\right)^2 \right\|_2} + 0.01 \cdot \left\| x^{(1)} + 10 \right\|_2$

   constraints: $\Omega = \{\boldsymbol{x} : \boldsymbol{l} \leq \boldsymbol{x} \leq \boldsymbol{u}\}$, $\Xi = \mathbb{R}^n$

   ▹ $\boldsymbol{l} = \begin{bmatrix} -15 & -5 \end{bmatrix}^\top, \boldsymbol{u} = \begin{bmatrix} -5 & 3 \end{bmatrix}^\top$

   minimizer: $\boldsymbol{x^*} = \begin{bmatrix} -10 & 1 \end{bmatrix}^\top$

   minimum: $f^* = 0$

5. `levi 13` [92]:

   dimensionality: $n = 2$

   cost function: $f(\boldsymbol{x}) = \sin\left(3 \cdot \pi \cdot x^{(1)}\right)^2 + \left(x^{(1)} - 1\right)^2 \cdot \left[1 + \sin\left(3 \cdot \pi \cdot x^{(2)}\right)^2\right] + + \left(x^{(2)} - 1\right)^2 \cdot \left[1 + \sin\left(2 \cdot \pi \cdot x^{(2)}\right)^2\right]$

   constraints: $\Omega = \{\boldsymbol{x} : \boldsymbol{l} \leq \boldsymbol{x} \leq \boldsymbol{u}\}$, $\Xi = \mathbb{R}^n$

   ▹ $\boldsymbol{l} = -10 \cdot \mathbf{1}_n, \boldsymbol{u} = 10 \cdot \mathbf{1}_n$

   minimizer: $\boldsymbol{x^*} = \mathbf{1}_n$

   minimum: $f^* = 0$

6. `adjiman` [62]:

   dimensionality: $n = 2$

   cost function: $f(\boldsymbol{x}) = \cos\left(x^{(1)}\right) \cdot \sin\left(x^{(2)}\right) - \frac{x^{(1)}}{\left(x^{(2)}\right)^2 + 1}$

   constraints: $\Omega = \{\boldsymbol{x} : \boldsymbol{l} \leq \boldsymbol{x} \leq \boldsymbol{u}\}$, $\Xi = \mathbb{R}^n$

   ▹ $\boldsymbol{l} = \begin{bmatrix} -1 & -1 \end{bmatrix}^\top, \boldsymbol{u} = \begin{bmatrix} 2 & 1 \end{bmatrix}^\top$

   minimizer: $\boldsymbol{x^*} = \begin{bmatrix} 2 & 0.10578 \end{bmatrix}^\top$

   minimum: $f^* = -2.02181$

7. `rosenbrock` [62]:

   dimensionality: $n = 5$

   cost function: $f(\boldsymbol{x}) = \sum_{i=1}^{n-1} \left\{100 \cdot \left[x^{(i+1)} - \left(x^{(i)}\right)^2\right]^2 + \left[x^{(i)} - 1\right]^2\right\}$

constraints: $\Omega = \{\boldsymbol{x} : \boldsymbol{l} \leq \boldsymbol{x} \leq \boldsymbol{u}\}, \Xi = \mathbb{R}^n$

▹ $\boldsymbol{l} = -30 \cdot \mathbf{1}_n, \boldsymbol{u} = 30 \cdot \mathbf{1}_n$

minimizer: $\boldsymbol{x}^* = \mathbf{1}_n$

minimum: $f^* = 0$

8. `step 2` [62]:

dimensionality: $n = 5$

cost function: $f(\boldsymbol{x}) = \sum_{i=1}^{n} \left\lfloor x^{(i)} + 0.5 \right\rfloor^2$

constraints: $\Omega = \{\boldsymbol{x} : \boldsymbol{l} \leq \boldsymbol{x} \leq \boldsymbol{u}\}, \Xi = \mathbb{R}^n$

▹ $\boldsymbol{l} = -100 \cdot \mathbf{1}_n, \boldsymbol{u} = 100 \cdot \mathbf{1}_n$

minimizer: $\boldsymbol{x}^* = -0.5 \cdot \mathbf{1}_n$

minimum: $f^* = 0$

9. `salomon` [62]:

dimensionality: $n = 5$

cost function: $f(\boldsymbol{x}) = 1 - \cos\left[2 \cdot \pi \cdot \sqrt{\sum_{i=1}^{n} \left(x^{(i)}\right)^2}\right] + 0.1 \cdot \sqrt{\sum_{i=1}^{n} \left(x^{(i)}\right)^2}$

constraints: $\Omega = \{\boldsymbol{x} : \boldsymbol{l} \leq \boldsymbol{x} \leq \boldsymbol{u}\}, \Xi = \mathbb{R}^n$

▹ $\boldsymbol{l} = -100 \cdot \mathbf{1}_n, \boldsymbol{u} = 100 \cdot \mathbf{1}_n$

minimizer: $\boldsymbol{x}^* = \mathbf{0}_n$

minimum: $f^* = 0$

B.2 Constrained optimization problems

1. `gramacy and lee constrained` (adapted from [53]):

dimensionality: $n = 1$

cost function: $f(x) = \frac{\sin\left(10 \cdot \pi \cdot x^{(1)}\right)}{2 \cdot x^{(1)}} + \left(x^{(1)} - 1\right)^4$

constraints: $\Omega = \{x : l \leq x \leq u\}, \Xi = \{x : g_{\Xi}(x) \leq 0\}$

▹ $l = 0.5, u = 2.5$

▹ $g_{\Xi}(x) = \sin\left[-2 \cdot \left(x^{(1)}\right)^3 + 8 \cdot x^{(1)} - 3 \cdot \left(x^{(1)}\right)^2\right]$

minimizer: $x^* = 0.5486$

minimum: $f^* = -0.8690$

2. `sasena 1` [125]:

   dimensionality: $n = 2$

   cost function: $f(\boldsymbol{x}) = 2 + \frac{\left[x^{(2)} - \left(x^{(1)}\right)^2\right]^2}{100} + \left[1 - x^{(1)}\right]^2 + 2 \cdot \left[2 - x^{(2)}\right]^2 + +7 \cdot \sin\left(\frac{x^{(1)}}{2}\right) \cdot \sin\left(\frac{7}{10} \cdot x^{(1)} \cdot x^{(2)}\right)$

   constraints: $\Omega = \{\boldsymbol{x} : \boldsymbol{l} \leq \boldsymbol{x} \leq \boldsymbol{u}\}$, $\Xi = \{\boldsymbol{x} : g_\Xi(\boldsymbol{x}) \leq 0\}$

   ▹ $\boldsymbol{l} = \boldsymbol{0}_n, \boldsymbol{u} = 5 \cdot \boldsymbol{1}_n$

   ▹ $g_\Xi(\boldsymbol{x}) = -\sin\left(x^{(1)} - x^{(2)} - \frac{\pi}{8}\right)$

   minimizer: $\boldsymbol{x}^* = \begin{bmatrix} 2.7450 & 2.3523 \end{bmatrix}^\top$

   minimum: $f^* = -1.1743$

3. `townsend` [1]:

   dimensionality: $n = 2$

   cost function: $f(\boldsymbol{x}) = -\left\{\cos\left[\left(x^{(1)} - 0.1\right) \cdot x^{(2)}\right]\right\}^2 - x^{(1)} \cdot \sin\left(3 \cdot x^{(1)} + x^{(2)}\right)$

   constraints: $\Omega = \{\boldsymbol{x} : \boldsymbol{l} \leq \boldsymbol{x} \leq \boldsymbol{u}\}$, $\Xi = \{\boldsymbol{x} : g_\Xi(\boldsymbol{x}) \leq 0\}$

   ▹ $\boldsymbol{l} = \begin{bmatrix} -2.25 & -2.5 \end{bmatrix}^\top, \boldsymbol{u} = \begin{bmatrix} 2.5 & 1.75 \end{bmatrix}^\top$

   ▹ $g_\Xi(\boldsymbol{x}) = \left(x^{(1)}\right)^2 + \left(x^{(2)}\right)^2 - \left[2 \cdot \cos(t) - \frac{\cos(2 \cdot t)}{2} - \frac{\cos(3 \cdot t)}{4} - \frac{\cos(4 \cdot t)}{8}\right]^2 - [2 \cdot \sin(t)]^2$, where $t = \text{arctan2}\left(x^{(1)}, x^{(2)}\right)$

   minimizer: $\boldsymbol{x}^* = \begin{bmatrix} 2.0052938 & 1.1944509 \end{bmatrix}^\top$

   minimum: $f^* = -2.0240$

4. `mishra's bird` [156]:

   dimensionality: $n = 2$

   cost function: $f(\boldsymbol{x}) = \sin\left(x^{(2)}\right) \cdot \exp\left\{\left[1 - \cos\left(x^{(1)}\right)\right]^2\right\} + \cos\left(x^{(1)}\right) \cdot \exp\left\{\left[1 - \sin\left(x^{(2)}\right)\right]^2\right\} + + \left[x^{(1)} - x^{(2)}\right]^2$

   constraints: $\Omega = \{\boldsymbol{x} : \boldsymbol{l} \leq \boldsymbol{x} \leq \boldsymbol{u}\}$, $\Xi = \{\boldsymbol{x} : g_\Xi(\boldsymbol{x}) \leq 0\}$

   ▹ $\boldsymbol{l} = \begin{bmatrix} -10 & -6.5 \end{bmatrix}^\top, \boldsymbol{u} = \begin{bmatrix} -2 & 0 \end{bmatrix}^\top$

   ▹ $g_\Xi(\boldsymbol{x}) = \left[x^{(1)} + 9\right]^2 + \left[x^{(2)} + 3\right]^2 - 9$

   minimizer: $\boldsymbol{x}^* = \begin{bmatrix} -9.367558 & -1.628040 \end{bmatrix}^\top$

minimum: $f^* = -48.4060$

5. `camel six humps constrained` [156]:

   dimensionality: $n = 2$

   cost function: $f(\boldsymbol{x}) = \left[4 - 2.1 \cdot \left(x^{(1)}\right)^2 + \frac{\left(x^{(1)}\right)^4}{3}\right] \cdot \left(x^{(1)}\right)^2 + x^{(1)} \cdot x^{(2)} + \left[4 \cdot \left(x^{(2)}\right)^2 - 4\right] \cdot \left(x^{(2)}\right)^2$

   constraints: $\Omega = \{\boldsymbol{x} : \boldsymbol{l} \leq \boldsymbol{x} \leq \boldsymbol{u}\}$, $\Xi = \left\{\boldsymbol{x} : g_\Xi^{(1)}(\boldsymbol{x}) \leq 0, A_\Xi \cdot \boldsymbol{x} \leq \boldsymbol{b}_\Xi\right\}$

   ▹ $\boldsymbol{l} = \begin{bmatrix} -2 & -1 \end{bmatrix}^\top, \boldsymbol{u} = \begin{bmatrix} 2 & 1 \end{bmatrix}^\top$

   ▹ $g_\Xi^{(1)}(\boldsymbol{x}) = \left(x^{(1)}\right)^2 + \left[x^{(2)} + 0.1\right]^2 - 0.5$

   ▹ $A_\Xi = \begin{bmatrix} 1.6295 & 1 \\ -1 & 4.4553 \\ -4.3023 & -1 \\ -5.6905 & -12.1374 \\ 17.6198 & 1 \end{bmatrix}$ and $\boldsymbol{b}_\Xi = \begin{bmatrix} 3.0786 \\ 2.7417 \\ -1.4909 \\ 1 \\ 32.5198 \end{bmatrix}$

   minimizer: $\boldsymbol{x}^* = \begin{bmatrix} 0.212640 & 0.575114 \end{bmatrix}^\top$

   minimum: $f^* = -0.5865$

6. `sasena 2` [125]:

   dimensionality: $n = 2$

   cost function: $f(\boldsymbol{x}) = -\left[x^{(1)} - 1\right]^2 - \left[x^{(2)} - 0.5\right]^2$

   constraints: $\Omega = \{\boldsymbol{x} : \boldsymbol{l} \leq \boldsymbol{x} \leq \boldsymbol{u}\}$, $\Xi = \left\{\boldsymbol{x} : \boldsymbol{g}_\Xi(\boldsymbol{x}) \leq \boldsymbol{0}_{q_\Xi}\right\}, q_\Xi = 3$

   ▹ $\boldsymbol{l} = \boldsymbol{0}_n, \boldsymbol{u} = \boldsymbol{1}_n$

   ▹ $g_\Xi^{(1)}(\boldsymbol{x}) = \left\{\left[x^{(1)} - 3\right]^2 + \left[x^{(2)} + 2\right]^2\right\} \cdot \exp\left\{-\left(x^{(2)}\right)^7\right\} - 12$

   ▹ $g_\Xi^{(2)}(\boldsymbol{x}) = 10 \cdot x^{(1)} + x^{(2)} - 7$

   ▹ $g_\Xi^{(3)}(\boldsymbol{x}) = \left[x^{(1)} - 0.5\right]^2 + \left[x^{(2)} - 0.5\right]^2 - 0.2$

   minimizer: $\boldsymbol{x}^* = \begin{bmatrix} 0.2017 & 0.8332 \end{bmatrix}$

   minimum: $f^* = -0.7483$

7. `welded beam design` [78]:

   dimensionality: $n = 4$

cost function: $f(\boldsymbol{x}) = 0.04811 \cdot x^{(3)} \cdot x^{(4)} \cdot \left[x^{(2)} + 14\right] + 1.10471 \cdot \left(x^{(1)}\right)^2 \cdot x^{(2)}$

constraints: $\Omega = \{\boldsymbol{x} : \boldsymbol{l} \leq \boldsymbol{x} \leq \boldsymbol{u}\}$, $\Xi = \left\{\boldsymbol{x} : \boldsymbol{g}_\Xi(\boldsymbol{x}) \leq \boldsymbol{0}_{q_\Xi}\right\}$, $q_\Xi = 5$

- ▹ $\boldsymbol{l} = \begin{bmatrix} 0.125 & 0.1 & 0.1 & 0.1 \end{bmatrix}^\top$, $\boldsymbol{u} = \begin{bmatrix} 2 & 10 & 10 & 2 \end{bmatrix}^\top$
- ▹ $g_\Xi^{(1)}(\boldsymbol{x}) = x^{(1)} - x^{(4)}$
- ▹ $g_\Xi^{(2)}(\boldsymbol{x}) = \delta(\boldsymbol{x}) - \delta_{max}$
- ▹ $g_\Xi^{(3)}(\boldsymbol{x}) = P - P_c(\boldsymbol{x})$
- ▹ $g_\Xi^{(4)}(\boldsymbol{x}) = \tau(\boldsymbol{x}) - \tau_{max}$
- ▹ $g_\Xi^{(5)}(\boldsymbol{x}) = \sigma(\boldsymbol{x}) - \sigma_{max}$
- ▹ $\tau(\boldsymbol{x}) = \sqrt{(\tau'(\boldsymbol{x}))^2 + (\tau''(\boldsymbol{x}))^2 + 2 \cdot \tau'(\boldsymbol{x}) \cdot \tau''(\boldsymbol{x}) \cdot \frac{x^{(2)}}{2 \cdot R(\boldsymbol{x})}}$
- ▹ $\tau'(\boldsymbol{x}) = \frac{P}{\sqrt{2} \cdot x^{(2)} \cdot x^{(1)}}$
- ▹ $\tau''(\boldsymbol{x}) = \frac{R(\boldsymbol{x}) \cdot M(\boldsymbol{x})}{J(\boldsymbol{x})}$
- ▹ $M(\boldsymbol{x}) = P \cdot \left(\frac{x^{(2)}}{2} + L\right)$
- ▹ $R(\boldsymbol{x}) = \sqrt{\frac{\left(x^{(2)}\right)^2}{4} + \left(\frac{x^{(1)} + x^{(3)}}{2}\right)^2}$
- ▹ $J(\boldsymbol{x}) = 2 \cdot \left[\left(\frac{\left(x^{(2)}\right)^2}{4} + \left(\frac{x^{(1)} + x^{(3)}}{2}\right)^2\right) \cdot \sqrt{2} \cdot x^{(1)} \cdot x^{(2)}\right]$
- ▹ $\sigma(\boldsymbol{x}) = \frac{6 \cdot P \cdot L}{x^{(4)} \cdot \left(x^{(3)}\right)^2}$
- ▹ $\delta(\boldsymbol{x}) = \frac{6 \cdot P \cdot L^3}{E \cdot \left(x^{(3)}\right)^2 \cdot x^{(4)}}$
- ▹ $P_c(\boldsymbol{x}) = \frac{4.013 \cdot E \cdot x^{(3)} \cdot \left(x^{(4)}\right)^3}{6 \cdot L^2} \cdot \left[1 - \frac{x^{(3)}}{2 \cdot L} \cdot \sqrt{\frac{E}{4 \cdot G}}\right]$
- ▹ $L = 14, P = 6000, E = 30 \cdot 10^6, \sigma_{max} = 30000, \tau_{max} = 13600, G = 12 \cdot 10^6, \sigma_{max} = 0.25$

minimizer: $\boldsymbol{x}^* = \begin{bmatrix} 0.20573 & 3.47049 & 9.03662 & 0.20573 \end{bmatrix}$ (taken from [70])

minimum: $f^* = 1.7249$

8. `himmelblau` [78]:

dimensionality: $n = 5$

cost function: $f(\boldsymbol{x}) = 5.3578547 \cdot \left(x^{(3)}\right)^2 + 0.8356891 \cdot x^{(1)} \cdot x^{(5)} + +37.293239 \cdot x^{(1)} - 40792.141$

constraints: $\Omega = \{\boldsymbol{x} : \boldsymbol{l} \leq \boldsymbol{x} \leq \boldsymbol{u}\}$, $\Xi = \left\{\boldsymbol{x} : \boldsymbol{g}_\Xi(\boldsymbol{x}) \leq \boldsymbol{0}_{q_\Xi}\right\}$, $q_\Xi = 6$

- ▹ $\boldsymbol{l} = \begin{bmatrix} 78 & 33 & 27 & 27 & 27 \end{bmatrix}^\top$, $\boldsymbol{u} = \begin{bmatrix} 102 & 45 & 45 & 45 & 45 \end{bmatrix}^\top$
- ▹ $g_\Xi^{(1)}(\boldsymbol{x}) = -G_1(\boldsymbol{x})$
- ▹ $g_\Xi^{(2)}(\boldsymbol{x}) = G_1(\boldsymbol{x}) - 92$
- ▹ $g_\Xi^{(3)}(\boldsymbol{x}) = 90 - G_2(\boldsymbol{x})$

- ▹ $g_{\Xi}^{(4)}(\boldsymbol{x}) = G_2(\boldsymbol{x}) - 110$
- ▹ $g_{\Xi}^{(5)}(\boldsymbol{x}) = 20 - G_3(\boldsymbol{x})$
- ▹ $g_{\Xi}^{(6)}(\boldsymbol{x}) = G_3(\boldsymbol{x}) - 25$
- ▹ $G_1(\boldsymbol{x}) = 85.334407 + 0.0056858 \cdot x^{(2)} \cdot x^{(5)} + 0.0006262 \cdot x^{(1)} \cdot x^{(4)} + -0.0022053 \cdot x^{(3)} \cdot x^{(5)}$
- ▹ $G_2(\boldsymbol{x}) = 80.51249 + 0.0071317 \cdot x^{(2)} \cdot x^{(5)} + 0.0029955 \cdot x^{(1)} \cdot x^{(2)} + +0.0021813 \cdot \left(x^{(3)}\right)^2$
- ▹ $G_3(\boldsymbol{x}) = 9.300961 + 0.0047026 \cdot x^{(3)} \cdot x^{(5)} + 0.00125447 \cdot x^{(1)} \cdot x^{(3)} + +0.0019085 \cdot x^{(3)} \cdot x^{(4)}$

minimizer: $\boldsymbol{x}^* = \begin{bmatrix} 78 & 33.002617891740300 & 30.023386693211926 & 45 & 36.712662729997280 \end{bmatrix}$ (found by `PSWARM` [72])

minimum: $f^* = -30661$

9. `step 2 constrained`:

dimensionality: $n = 5$

cost function: $f(\boldsymbol{x}) = \sum_{i=1}^{n} \left\lfloor x^{(i)} + 0.5 \right\rfloor^2$

constraints: $\Omega = \{\boldsymbol{x} : \boldsymbol{l} \leq \boldsymbol{x} \leq \boldsymbol{u}\}$, $\Xi = \left\{\boldsymbol{x} : \boldsymbol{g}_{\Xi}(\boldsymbol{x}) \leq \boldsymbol{0}_{q_{\Xi}}\right\}$, $q_{\Xi} = n + 1$

- ▹ $\boldsymbol{l} = -100 \cdot \boldsymbol{1}_n, \boldsymbol{u} = 100 \cdot \boldsymbol{1}_n$
- ▹ $g_{\Xi}^{(j)}(\boldsymbol{x}) = x^{(j)} + 0.5, j = 1, \ldots, n$
- ▹ $g_{\Xi}^{(n+1)}(\boldsymbol{x}) = \|\boldsymbol{x} - \boldsymbol{x_c}\|_2^2 - r^2$, where $\boldsymbol{x_c} = \boldsymbol{0}_n$ and $r = \frac{3}{8} \cdot \|\boldsymbol{u} - \boldsymbol{l}\|_2$

minimizer: $\boldsymbol{x}^* = -0.5 \cdot \boldsymbol{1}_n$

minimum: $f^* = 0$

Appendix C. Hyper-parameters for the procedures

This Appendix reports the hyper-parameters and settings used by the procedures benchmarked in Chapter 7, namely `GLIS` [10], `GLISp` [11], `C-GLIS`, `C-GLISp` [156], `GLIS-r` [108], `GLISp-r` [109], `C-GLIS-r` and `C-GLISp-r`.

Table 9: Settings for `GLIS` [10], `C-GLIS` and `GLIS-r` [108] for the unconstrained black-box optimization benchmarks in Chapter 7. The acronym `n.d.` indicates that the hyper-parameter is not defined for one of the methods, while / highlights that the hyper-parameter has not been used.

	`GLIS` [10] and `C-GLIS`	`GLIS-r` [108]	`GLIS-r` [108] exploitation	`GLIS-r` [108] exploration
Radial function $\varphi_f(\cdot)$ for $\hat{f}_N(\boldsymbol{x})$	inverse quadratic			
Shape parameter ϵ_f for $\hat{f}_N(\boldsymbol{x})$	$1.0755/n$			
Threshold ϵ_{SVD}	10^{-6}			
IDW distance weight δ_1	$1.4246/n$	`n.d.`		
IDW variance weight δ_2	$1.5078/n$	`n.d.`		
Safeguard threshold $\epsilon_{\Delta Y}$	10^{-5}	`n.d.`		
Cycling set Δ_{cycle}	`n.d.`	$\langle 0.95, 0.7, 0.35, 0 \rangle$	$\langle 0.95 \rangle$	$\langle 0 \rangle$
Number of centroids K_{aug}	`n.d.`	5		
Recalibration indexes $\mathcal{K}_{R_f}$ for $\hat{f}_N(\boldsymbol{x})$	$\emptyset$ (no recalibration)			
Grid of ϵ_f for cross-validation	/			
Folds ratio for cross-validation of ϵ_f: R_f	/			
Number of initial samples N_{init}	$2 \cdot n$			
Budget N_{max}	200			
Global optimization solver	`PSWARM` [72]			

Table 10: Settings for `GLISp` [11], `C-GLISp` [156] and `GLISp-r` [109] for the unconstrained preference-based optimization benchmarks in Chapter 7. The acronym `n.d.` indicates that the hyper-parameter is not defined for one of the methods, while / highlights that the hyper-parameter has not been used.

	`GLISp` [11] and `C-GLISp` [156]	`GLISp-r` [109]	`GLISp-r` [109] exploitation	`GLISp-r` [109] exploration
Radial function $\varphi_f(\cdot)$ for $\hat{f}_N(\boldsymbol{x})$	inverse quadratic			
Shape parameter ϵ_f for $\hat{f}_N(\boldsymbol{x})$	1			
Regularization parameter λ_f for $\hat{f}_N(\boldsymbol{x})$	10^{-6}			
Tolerance σ_π for $\hat{f}_N(\boldsymbol{x})$	10^{-2}			
IDW distance weight δ	2	`n.d.`		
Cycling set Δ_{cycle}	`n.d.`	$\langle 0.95, 0.7, 0.35, 0 \rangle$	$\langle 0.95 \rangle$	$\langle 0 \rangle$
Number of centroids K_{aug}	`n.d.`	5		
Recalibration indexes $\mathcal{K}_{R_f}$ for $\hat{f}_N(\boldsymbol{x})$	$\{1, 50, 100\}$			$\emptyset$
Grid of ϵ_f for cross-validation	$\{0.1000, 0.1668, 0.2783, 0.4642, 0.7743,$ $1.0000, 1.2915, 2.1544, 3.5938, 5.9948, 10\}$			$\emptyset$
Folds ratio for cross-validation of ϵ_f: R_f	LOOCV $(R_f = 0)$			$\emptyset$
Number of initial samples N_{init}	$4 \cdot n$			
Budget N_{max}	200			
Global optimization solver	`PSWARM` [72]			

Table 11: Settings for `C-GLIS` and `C-GLIS-r` for the constrained black-box optimization benchmarks in Chapter 7. The acronym `n.d.` indicates that the hyper-parameter is not defined for one of the methods, while / highlights that the hyper-parameter has not been used.

	`C-GLIS`	`C-GLIS-r` PSVM	`C-GLIS-r` IDWI
Radial function $\varphi_f(\cdot)$ for $\hat{f}_N(\boldsymbol{x})$	inverse quadratic		
Shape parameter ϵ_f for $\hat{f}_N(\boldsymbol{x})$	$1.0755/n$		
Threshold ϵ_{SVD}	10^{-6}		
Radial function $\varphi_\Xi(\cdot)$ for $\hat{u}_{\Xi_N}(\boldsymbol{x})$	`n.d.`	Gaussian	`n.d.`
Shape parameter ϵ_Ξ for $\hat{u}_{\Xi_N}(\boldsymbol{x})$	`n.d.`	1	`n.d.`
Trade-off parameter for PSVM: C_{SVM}	`n.d.`	10^6	`n.d.`
Threshold γ for $\hat{u}_{\Xi_N}(\boldsymbol{x})$	`n.d.`	0.5	
IDW distance weight δ_1	2	`n.d.`	
IDW variance weight δ_2	0.4	`n.d.`	
Default penalty weight $\delta_{\Xi,default}$	2	`n.d.`	
Safeguard threshold $\epsilon_{\Delta Y}$	10^{-5}	`n.d.`	
Cycling set Δ_{cycle}	`n.d.`	$\langle 0.95, 0.7, 0.35, 0 \rangle$	
Number of centroids K_{aug}	`n.d.`	5	
Recalibration indexes $\mathcal{K}_{R_f}$ for $\hat{f}_N(\boldsymbol{x})$	$\emptyset$ (no recalibration)		
Grid of ϵ_f for cross-validation	/		
Folds ratio for cross-validation of ϵ_f: R_f	/		
Recalibration indexes $\mathcal{K}_{R_\Xi}$ for $\hat{u}_{\Xi_N}(\boldsymbol{x})$	`n.d.`	$\emptyset$ (no recalibration)	`n.d.`
Bounds for PSVM hyper-parameters: $\boldsymbol{l}_\Xi$ and $\boldsymbol{u}_\Xi$	`n.d.`	/	`n.d.`
Folds ratio for cross-validation of ϵ_Ξ, C_{SVM}: R_Ξ	`n.d.`	/	`n.d.`
Number of initial samples N_{init}	$6 \cdot n$		
Budget N_{max}	200		
Global optimization solver	`PSWARM` [72]		

Table 12: Settings for `C-GLISp` [156] and `C-GLISp-r` for the constrained preference-based optimization benchmarks in Chapter 7. The acronym `n.d.` indicates that the hyper-parameter is not defined for one of the methods, while / highlights that the hyper-parameter has not been used.

	`C-GLISp` [156]	`C-GLISp-r` PSVM	`C-GLISp-r` IDWI
Radial function $\varphi_f(\cdot)$ for $\hat{f}_N(\boldsymbol{x})$	inverse quadratic	inverse quadratic	inverse quadratic
Shape parameter ϵ_f for $\hat{f}_N(\boldsymbol{x})$	1	1	1
Regularization parameter λ_f for $\hat{f}_N(\boldsymbol{x})$	10^{-6}	10^{-6}	10^{-6}
Tolerance σ_π for $\hat{f}_N(\boldsymbol{x})$	10^{-2}	10^{-2}	10^{-2}
Radial function $\varphi_\Xi(\cdot)$ for $\hat{u}_{\Xi_N}(\boldsymbol{x})$	`n.d.`	Gaussian	`n.d.`
Shape parameter ϵ_Ξ for $\hat{u}_{\Xi_N}(\boldsymbol{x})$	`n.d.`	1	`n.d.`
Trade-off parameter for PSVM: C_{SVM}	`n.d.`	10^6	`n.d.`
Threshold γ for $\hat{u}_{\Xi_N}(\boldsymbol{x})$	`n.d.`	0.5	0.5
IDW distance weight δ	2	`n.d.`	`n.d.`
Default penalty weight $\delta_{\Xi,default}$	2	`n.d.`	`n.d.`
Exploration-exploitation cycle Δ_{cycle}	`n.d.`	$\langle 0.95, 0.7, 0.35, 0\rangle$	$\langle 0.95, 0.7, 0.35, 0\rangle$
Number of centroids K_{aug}	`n.d.`	5	5
Recalibration indexes $\mathcal{K}_{R_f}$ for $\hat{f}_N(\boldsymbol{x})$	$\{1, 50, 100\}$	$\{1, 50, 100\}$	$\{1, 50, 100\}$
Grid of ϵ_f for cross-validation	$\{0.1000, 0.1668, 0.2783, 0.4642, 0.7743, 1.0000, 1.2915, 2.1544, 3.5938, 5.9948, 10\}$	$\{0.1000, 0.1668, 0.2783, 0.4642, 0.7743, 1.0000, 1.2915, 2.1544, 3.5938, 5.9948, 10\}$	$\{0.1000, 0.1668, 0.2783, 0.4642, 0.7743, 1.0000, 1.2915, 2.1544, 3.5938, 5.9948, 10\}$
Folds ratio for cross-validation of ϵ_f: R_f	LOOCV $(R_f = 0)$	LOOCV $(R_f = 0)$	LOOCV $(R_f = 0)$
Recalibration indexes $\mathcal{K}_{R_\Xi}$ for $\hat{u}_{\Xi_N}(\boldsymbol{x})$	`n.d.`	$\emptyset$ (no recalibration)	`n.d.`
Bounds for PSVM hyper-parameters: $\boldsymbol{l}_\Xi$ and $\boldsymbol{u}_\Xi$	`n.d.`	/	`n.d.`
Folds ratio for cross-validation of ϵ_Ξ, C_{SVM}: R_Ξ	`n.d.`	/	`n.d.`
Number of initial samples N_{init}	$6 \cdot n$	$6 \cdot n$	$6 \cdot n$
Budget N_{max}	200	200	200
Global optimization solver	`PSWARM` [72]	`PSWARM` [72]	`PSWARM` [72]

List of Notations

Important sets of numbers

- $\mathbb{N}$ is the set of all natural numbers (zero not included);
- $\mathbb{Z}$ is the set of all integers;
- $\mathbb{Q}$ is the set of all rational numbers;
- $\mathbb{R}$ is the set of all real numbers;
- $\mathbb{R}_{\geq 0}$ is the set of all non-negative real numbers;
- $\mathbb{R}_{>0}$ is the set of all positive real numbers;
- $\mathbb{R}^n$, $n \in \mathbb{N}$, is the n-dimensional Euclidean space.

Notation of other sets

- Generic sets are indicated with upper-case calligraphic letters, e.g. $\mathcal{A}$;
- $\emptyset$ denotes the empty set;
- Given a set $\mathcal{A}$, $|\mathcal{A}|$ is its cardinality;
- Generic classes (i.e. nonempty collections of sets) are also indicated with upper-case calligraphic letters, e.g. $\mathcal{C}$;
- Given a class $\mathcal{C}$, we refer to the i-th set contained in $\mathcal{C}$ as $\mathcal{C}^{(i)}$;
- $\mathcal{N}(\tilde{\boldsymbol{x}})$ is a neighborhood of a point $\tilde{\boldsymbol{x}} \in \mathbb{R}^n$;
- $\mathcal{B}(\tilde{\boldsymbol{x}}; \epsilon)$ is an open ball of radius $\epsilon \in \mathbb{R}_{>0}$ around a point $\tilde{\boldsymbol{x}} \in \mathbb{R}^n$. Instead, $\bar{\mathcal{B}}(\tilde{\boldsymbol{x}}; \epsilon)$ is a closed ball of radius $\epsilon \in \mathbb{R}_{>0}$ around $\tilde{\boldsymbol{x}} \in \mathbb{R}^n$.

Vectors and matrices

Let $n, m \in \mathbb{N}$, then:

- Apart from very few exceptions, generic scalars are indicated with lower-case letters, e.g. a;

- Generic vectors are indicated with lower-case bold letters, e.g. $\boldsymbol{x}$;
- Generic matrices are indicated with upper-case letters, e.g. A;
- All vectors are to be treated as column vectors, e.g. $\boldsymbol{x} \in \mathbb{R}^n$ denotes an n-dimensional column vector of real numbers;
- $x^{(j)} \in \mathbb{R}$ is the j-th entry of vector $\boldsymbol{x} \in \mathbb{R}^n$;
- $A^{(i,j)} \in \mathbb{R}$ is the (i, j)-th element of matrix $A \in \mathbb{R}^{n \times m}$;
- $\mathbf{0}_n$ is an n-dimensional column vector of zeros;
- $0_{n \times m}$ is an $(n \times m)$-dimensional matrix of zeros;
- $\mathbf{1}_n$ is an n-dimensional column vector of ones;
- $1_{n \times m}$ is an $(n \times m)$-dimensional matrix of ones;
- $\mathbb{R}^{n \times m}$ is the set of all real matrices with n rows and m columns;
- I_n is the $n \times n$ identity matrix;
- $\boldsymbol{e}_j$ is the j-th column of the identity matrix. Its dimensionality is stated explicitly whenever it is not clear from the context;
- The transpose of a matrix $A \in \mathbb{R}^{n \times m}$ is $A^\top$ and similarly for vectors;
- The inverse of an invertible square matrix $A \in \mathbb{R}^{n \times n}$ is A^{-1};
- The determinant of a square matrix $A \in \mathbb{R}^{n \times n}$ is $\det A$;
- $A = \operatorname{diag}\{a_1, \ldots, a_N\}$, $A \in \mathbb{R}^{n \times n}$, is a diagonal matrix with entries $a_i \in \mathbb{R}, i = 1, \ldots, n$, on the main diagonal;
- Consider two matrices $A, B \in \mathbb{R}^{n \times m}$, we denote their Hadamard product $C \in \mathbb{R}^{n \times m}$ as $C = A \odot B$, where $C^{(i,j)} = A^{(i,j)} \cdot B^{(i,j)}$.

Functions

Let $n, m \in \mathbb{N}$, then:

- Apart from very few exceptions, generic scalar functions are indicated with lower-case letters, e.g. $h : \mathbb{R}^n \to \mathbb{R}$;

- Generic vector-valued functions are indicated with bold lower-case letters, e.g. $\boldsymbol{h} : \mathbb{R}^n \rightarrow \mathbb{R}^m$;
- When referring to a function $h : \mathbb{R}^n \rightarrow \mathbb{R}$ "as a whole", we either use the notation $h(\cdot)$ or $h(\boldsymbol{x})$ (where $\boldsymbol{x}$ is a generic argument for the function), as opposed to simply h. Similarly for vector-valued functions;
- Consider a vector-valued function $\boldsymbol{h}(\boldsymbol{x}), \boldsymbol{h} : \mathbb{R}^n \rightarrow \mathbb{R}^m$, we denote the j-th function that composes it as $h^{(j)}(\boldsymbol{x})$, where $h^{(j)} : \mathbb{R}^n \rightarrow \mathbb{R}$;
- Consider a scalar function $h(\boldsymbol{x}), h : \mathbb{R}^n \rightarrow \mathbb{R}$:
  - $\frac{\partial}{\partial x^{(j)}} h(\boldsymbol{x}), \frac{\partial}{\partial x^{(j)}} h : \mathbb{R}^n \rightarrow \mathbb{R}$, is the partial derivative of $h(\boldsymbol{x})$ with respect to $x^{(j)}$;
  - $\nabla_{\boldsymbol{x}} h(\boldsymbol{x}), \nabla_{\boldsymbol{x}} h : \mathbb{R}^n \rightarrow \mathbb{R}^n$, is the gradient of $h(\boldsymbol{x})$;
  - $\nabla^2_{\boldsymbol{xx}} h(\boldsymbol{x}), \nabla^2_{\boldsymbol{xx}} h : \mathbb{R}^n \rightarrow \mathbb{R}^{n \times n}$, is the Hessian of $h(\boldsymbol{x})$.
- Let $h(\boldsymbol{x})$ be a scalar function, $h : \mathbb{R}^n \rightarrow \mathbb{R}$, and $\mathcal{A}$ be a subset of $\mathbb{R}^n$. We denote the range assumed by $h(\boldsymbol{x})$ over $\mathcal{A}$ as $h[\mathcal{A}]$, i.e.

  $$h[\mathcal{A}] = \left\{ \tilde{h} : \tilde{h} \in \mathbb{R}, \tilde{h} = h(\tilde{\boldsymbol{x}}) \text{ for some } \tilde{\boldsymbol{x}} \in \mathcal{A} \right\},$$

  and similarly for vector-valued functions;
- Let $k \in \mathbb{N} \cup \{0\}$, we denote the class of continuously differentiable functions of order k on the open set $\mathcal{A} \subseteq \mathbb{R}^n$ as $C^k(\mathcal{A})$;
- Whenever needed, if a function $h(\cdot)$ depends on some additional parameters $\boldsymbol{\theta}$, we use a semicolon notation to separate the parameters $\boldsymbol{\theta}$ from the arguments $\boldsymbol{x}$ as follows: $h(\boldsymbol{x}; \boldsymbol{\theta})$.

Statistics

Let $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$ be two (continuous) random vectors, then:

- $p(\boldsymbol{x})$ is the (marginal) probability density function of $\boldsymbol{x}$;
- $p(\boldsymbol{x}, \boldsymbol{y})$ is the joint probability density function of $\boldsymbol{x}$ and $\boldsymbol{y}$;
- $p(\boldsymbol{x} \,|\, \boldsymbol{y})$ is the conditional probability density function of $\boldsymbol{x}$ given $\boldsymbol{y}$;
- $\mathbb{E}[\cdot]$ is the expected value operator;
- The notation $\boldsymbol{x} \sim \mathcal{P}$ is used to indicate that $\boldsymbol{x}$ is distributed according to some probability distribution $\mathcal{P}$;

- Consider a sequence of random vectors $\langle \boldsymbol{x}_1, \dots, \boldsymbol{x}_N \rangle, N \in \mathbb{N}$, we use the notation $\boldsymbol{x}_i \overset{i.i.d.}{\sim} \mathcal{P}$ to indicate that all the terms of the sequence are independent and identically distributed with distribution $\mathcal{P}$;

- $\mathcal{N}(\boldsymbol{\mu}, \Sigma)$ is the Gaussian (or normal) probability distribution with mean $\boldsymbol{\mu} \in \mathbb{R}^n$ and covariance matrix $\Sigma \in \mathbb{R}^{n \times n}$;

- $\phi_{\mathcal{N}}(\tilde{\boldsymbol{x}})$ and $\Phi_{\mathcal{N}}(\tilde{\boldsymbol{x}})$ denote, respectively, the standard normal probability distribution and the standard normal cumulative distribution evaluated at a point $\tilde{\boldsymbol{x}} \in \mathbb{R}^n$;

- $\mathcal{U}(\boldsymbol{l}, \boldsymbol{u})$ is the (continuous) multivariate uniform distribution with bounds $\boldsymbol{l}, \boldsymbol{u} \in \mathbb{R}^n$ and $\boldsymbol{l} \leq \boldsymbol{u}$.

Optimization

We use the following notation for global, black-box and preference-based optimization:

- $n \in \mathbb{N}$ is the dimensionality of the optimization problem;

- $\boldsymbol{x} \in \mathbb{R}^n$ is the decision vector;

- $x^{(j)}$ is the j-th decision variable;

- $f : \mathbb{R}^n \to \mathbb{R}$ is the cost function (to be minimized);

- $\gtrsim$ is the preference relation associated to a human decision-maker;

- Ω is a set of constraints that is a-priori known;

- Ξ is a set of constraints for which no mathematical formulation is available;

- $\boldsymbol{x}^+$ is a local solution for the optimization problem;

- $\boldsymbol{x}^*$ or $\boldsymbol{x}_i^*$ is a global solution for the optimization problem;

- $\mathcal{X}^*$ is the set containing all the global minimizers of the optimization problem;

- f^* is the global minimum of the optimization problem;

- k is the iteration counter of a generic optimization algorithm;

- N is the number of evaluated samples (at a given iteration of an optimization procedure);

- $\boldsymbol{x_{best}} \in \mathbb{R}^n$ is the best solution found by an optimization procedure. In particular, $\boldsymbol{x_{best}}(k) \in \mathbb{R}^n$ and $\boldsymbol{x_{best}}(N) \in \mathbb{R}^n$ denote, respectively, the best candidate sample after k iterations or when N points have been evaluated;

- $y_{best}, y_{best}(k), y_{best}(N) \in \mathbb{R}$ denote, respectively, the best cost, the best cost after k iterations or when N points have been evaluated by an optimization procedure.

Other notations

- $\|\cdot\|$ is a generic norm;

- $\|\cdot\|_p$ is the p-norm in $\mathbb{R}^n$;

- $\|\cdot\|_2$ is the Euclidean norm in $\mathbb{R}^n$;

- $\langle \boldsymbol{x}_i \rangle_{i \geq 1}$ is an infinite sequence of points (from $i = 1$ to ∞);

- $\langle \boldsymbol{x}_i \rangle_{i=1}^{k}$ is a finite sequence of points (from $i = 1$ to $k \in \mathbb{N}$);

- $\mathbb{I}(condition)$ is an indicator function which assumes value one whenever $condition$ is true and zero otherwise;

- Given $x \in \mathbb{R}$, $\lfloor x \rfloor$ denotes the greatest integer less than or equal to x. Instead, $\lceil x \rceil$ is the lowest integer greater than or equal to x;

- Given $x \in \mathbb{R}$, $\text{sign}\{x\}$ denotes the sign of x: $\text{sign}\{x\} = 1$ if $x > 0$, $\text{sign}\{x\} = -1$ if $x < 0$ and $\text{sign}\{x\} = 0$ if $x = 0$;

- Acronyms for algorithms are reported in typewriter font, followed by a reference (if available), e.g. `GS` [5];

- Benchmark optimization problems are also reported in typewriter font, followed by a reference (if available), e.g. `rosenbrock` [62].

List of Acronyms

$\mathcal{GP}$	Gaussian Process.
C-GLIS-r	Extension of GLIS-r [108] for constrained BBO proposed in this book.
C-GLISp-r	Extension of GLISp-r [109] for constrained PBO proposed in this book.
C-GLISp [156]	GLobal minimum using Inverse distance weighting and Surrogate radial basis functions for constrained PBO.
C-GLIS	GLobal minimum using Inverse distance weighting and Surrogate radial basis functions for constrained BBO.
COBRA [112]	Constrained Optimization By RAdial basis function interpolation.
CORS-RBF [115]	CORS [115] based on a RBF surrogate.
CORS [115]	Constrained Optimization using Response Surfaces.
ConstrLMSRBF [111]	Constrained Local MSRBF [116].
DIRECT [67]	DIvide a hyper-RECTangle.
EGO [68]	Efficient Global Optimization.
GLIS-r [108]	Extension of GLIS [10], based on min-max rescaling (r), proposed in this book.
GLISp-r [109]	Extension of GLISp [11], based on min-max rescaling (r), proposed in this book.
GLISp [11]	GLobal minimum using Inverse distance weighting and Surrogate radial basis functions for unconstrained PBO.
GLIS [10]	GLobal minimum using Inverse distance weighting and Surrogate radial basis functions.
GS [5]	Grid Search.

`Gutmann-RBF` [54]	Black-box optimization method based on RBFs proposed in [54].
`MSRBF` [116]	`MSRS` [116] based on a RBF surrogate.
`MSRS` [116]	Metric Stochastic Response Surface method.
`PSWARM` [72]	Particle SWARM.
`SO-SA` [151]	Extension of `MSRS` [116] proposed in [151].
`SuperEGO` [125]	Extension of the `EGO` [68] algorithm proposed in [125].
`gMRS` [108]	generalized Metric Response Surface.
BayesOpt	Bayesian Optimization.
BBO	Black-Box Optimization.
DM	Decision-Maker.
FF	Feed Forward (action).
GO	Global Optimization.
GOP	Global Optimization Problem.
IDW	Inverse Distance Weighting.
IDWI	Inverse Distance Weighting Interpolation.
KKT conditions	Karush-Kuhn-Tucker conditions.
LHD	Latin Hypercube Design.
LICQ	Linear Independence Constraint Qualification.
LOOCV	Leave-One-Out Cross-Validation.
LP	Linear Program.
MLE	Maximum Likelihood Estimation.
MOO	Multi-Objective Optimization.

PBO	Preference-Based Optimization.
PI	Proportional-Integral (controller).
PID	Proportional-Integral-Derivative (controller).
PrefBayesOpt	Preferential Bayesian Optimization.
PSVM	Probabilistic Support Vector Machine.
QP	Quadratic Program.
RBF	Radial Basis Function.
RMSE	Root Mean Square Error.
SVD	Singular Value Decomposition.
SVM	Support Vector Machine.

List of Figures

List of Tables

List of Algorithms

References

[1] Constrained optimization in chebfun. `http://www.chebfun.org/examples/opt/ConstrainedOptimization.html`. Accessed: 2022-08-08.

[2] Stéphane Alarie, Charles Audet, Aïmen E. Gheribi, Michael Kokkolaras, and Sébastien Le Digabel. Two decades of blackbox optimization applications. *EURO Journal on Computational Optimization*, 9:100011, 2021. doi: https://doi.org/10.1016/j.ejco.2021.100011.

[3] Candelieri Antonio. Sequential model based optimization of partially defined functions under unknown constraints. *Journal of Global Optimization*, 79(2):281–303, 2021. doi: https://doi.org/10.1007/s10898-019-00860-4.

[4] Karl Johan Åström and Tore Hägglund. *PID controllers: theory, design, and tuning*. ISA-The Instrumentation, Systems and Automation Society, 1995. ISBN 9781556175169.

[5] Charles Audet and Warren Hare. *Derivative-free and blackbox optimization*. Springer, 2017. ISBN 9783319886800.

[6] Charles Audet, Gilles Caporossi, and Stephane Jacquet. Binary, unrelaxable and hidden constraints in blackbox optimization. *Operations Research Letters*, 48(4):467–471, 2020. ISSN 0167-6377. doi: https://doi.org/10.1016/j.orl.2020.05.011.

[7] Robert J. Aumann. Utility theory without the completeness axiom. *Econometrica: Journal of the Econometric Society*, pages 445–462, 1962. doi: https://doi.org/10.2307/1909888.

[8] Thomas Back. *Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms*. Oxford university press, 1996. ISBN 9780195099713.

[9] Anirban Basudhar, Christoph Dribusch, Sylvain Lacaze, and Samy Missoum. Constrained efficient global optimization with support vector machines. *Structural and Multidisciplinary Optimization*, 46(2):201–221, 2012. doi: https://doi.org/10.1007/s00158-011-0745-5.

[10] Alberto Bemporad. Global optimization via inverse distance weighting and radial basis functions. *Computational Optimization and Applications*, 77(2):571–595, Nov 2020. ISSN 1573-2894. doi: https://doi.org/10.1007/s10589-020-00215-w.

[11] Alberto Bemporad and Dario Piga. Global optimization based on active preference learning with radial basis functions. *Machine Learning*, 110(2):417–448, Feb 2021. ISSN 1573-0565. doi: https://doi.org/10.1007/s10994-020-05935-y.

[12] Alessio Benavoli, Dario Azzimonti, and Dario Piga. Preferential bayesian optimisation with skew gaussian processes. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pages 1842–1850, 2021. doi: https://doi.org/10.1145/3449726.3463128.

[13] Viktor Bengs, Róbert Busa-Fekete, Adil El Mesaoudi-Paul, and Eyke Hüllermeier. Preference-based online learning with dueling bandits: A survey. *Journal of Machine Learning Research*, 22:7–1, 2021. URL `https://dl.acm.org/doi/10.5555/3546258.3546265`.

[14] Felix Berkenkamp, Angela P. Schoellig, and Andreas Krause. Safe controller optimization for quadrotors with gaussian processes. *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 491–496, 2016. doi: https://doi.org/10.1109/ICRA.2016.7487170.

[15] Christopher M. Bishop and Nasser M. Nasrabadi. *Pattern recognition and machine learning*. Springer, 2006. ISBN 9780387310732.

[16] Mattias Björkman and Kenneth Holmström. Global optimization of costly nonconvex functions using radial basis functions. *Optimization and Engineering*, 1(4):373–397, 2000. doi: https://doi.org/10.1023/A:1011584207202.

[17] Laurens Bliek, Arthur Guijt, Sicco Verwer, and Mathijs De Weerdt. Black-box mixed-variable optimisation using a surrogate model that satisfies integer constraints. *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pages 1851–1859, 2021. doi: https://doi.org/10.1145/3449726.3463136.

[18] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004. ISBN 0521833787.

[19] John S. Bradley, R. Reich, and S. G. Norcross. A just noticeable difference in c50 for speech. *Applied Acoustics*, 58(2):99–108, 1999. doi: https://doi.org/10.1016/S0003-682X(98)00075-9.

[20] Leo Breiman, Jerome H. Friedman, Richard A. Olshen, and Charles J. Stone. *Classification and regression trees*. Chapman and Hall/CRC, 1984. ISBN 9780412048418.

[21] Eric Brochu, Nando De Freitas, and Abhijeet Ghosh. Active preference learning with discrete choice data. *Advances in Neural Information Processing Systems 20 (NIPS*

2007), pages 409–416, 2007. URL `https://papers.nips.cc/paper/2007/hash/b6a1085a27ab7bff7550f8a3bd017df8-Abstract.html`.

[22] Eric Brochu, Vlad M. Cora, and Nando De Freitas. A tutorial on bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. *arXiv preprint arXiv:1012.2599*, 2010. doi: https://doi.org/10.48550/arXiv.1012.2599.

[23] Martin D. Buhmann. *Radial basis functions: theory and implementations*, volume 12. Cambridge university press, 2003. ISBN 9780511543241.

[24] Roberto Cavoretto, Alessandra De Rossi, Marat S. Mukhametzhanov, and Ya D. Sergeyev. On the search of the shape parameter in radial basis functions using univariate global optimization methods. *Journal of Global Optimization*, 79(2):305–327, 2021. doi: https://doi.org/10.1007/s10898-019-00853-3.

[25] Wei Chu and Zoubin Ghahramani. Preference learning with gaussian processes. *Proceedings of the 22nd international conference on Machine learning*, pages 137–144, 2005. doi: https://doi.org/10.1145/1102351.1102369.

[26] Alberto L. Cologni, Mirko Mazzoleni, and Fabio Previdi. Modeling and identification of an electro-hydraulic actuator. *2016 12th IEEE International Conference on Control and Automation (ICCA)*, pages 335–340, 2016. doi: https://doi.org/10.1109/ICCA.2016.7505299.

[27] Alberto Costa and Giacomo Nannicini. Rbfopt: an open-source library for black-box optimization with costly function evaluations. *Mathematical Programming Computation*, 10(4):597–629, 2018. doi: https://doi.org/10.1007/s12532-018-0144-7.

[28] Dennis D. Cox and Susan John. A statistical method for global optimization. In *Proceedings of the 1992 IEEE International Conference on Systems, Man, and Cybernetics*, pages 1241–1246. IEEE, 1992. doi: https://doi.org/10.1109/ICSMC.1992.271617.

[29] Nello Cristianini and John Shawe-Taylor. *An introduction to support vector machines and other kernel-based learning methods*. Cambridge university press, 2000. ISBN 9780521780193.

[30] Nello Cristianini and John Shawe-Taylor. *Support Vector Machines and other kernel-based learning methods*. Cambridge, 2004. ISBN 9780521780193.

[31] Gerard Debreu. *Theory of value: An axiomatic analysis of economic equilibrium*, volume 17. Yale University Press, 1971. ISBN 0300015593.

[32] Gianni Di Pillo and Luigi Grippo. Exact penalty functions in constrained optimization. *SIAM Journal on control and optimization*, 27(6):1333–1360, 1989. doi: https://doi.org/10.1137/0327068.

[33] Gianni Di Pillo, Stefano Lucidi, and Francesco Rinaldi. An approach to constrained global optimization based on exact penalty functions. *Journal of Global Optimization*, 54(2):251–260, 2012. doi: https://doi.org/10.1007/s10898-010-9582-0.

[34] Gianni Di Pillo, Giampaolo Liuzzi, Stefano Lucidi, Veronica Piccialli, and Francesco Rinaldi. A direct-type approach for derivative-free constrained global optimization. *Computational Optimization and Applications*, 65(2):361–397, 2016. doi: https://doi.org/10.1007/s10589-016-9876-3.

[35] Sébastien Le Digabel and Stefan M. Wild. A taxonomy of constraints in simulation-based optimization. *arXiv preprint arXiv:1505.07881*, 2015. doi: https://doi.org/10.48550/arXiv.1505.07881.

[36] Paweł D. Domański. *Control Performance Assessment: Theoretical Analyses and Industrial Practice*. Springer, 2020. ISBN 9783030235925.

[37] Russ Eberhart, Pat Simpson, and Roy Dobbins. *Computational intelligence PC tools*. Academic Press Professional, Inc., 1996. ISBN 9780122286308.

[38] Gregory E. Fasshauer. *Meshfree approximation methods with MATLAB*, volume 6. World Scientific, 2007. ISBN 9789812706348.

[39] Allan M. Feldman and Roberto Serrano. *Welfare economics and social choice theory*. Springer Science & Business Media, 2006. ISBN 9780387293677.

[40] Peter C. Fishburn. *Utility theory for decision making*. John Wiley & Sons, 1970. ISBN 9780471260608.

[41] Ronald Aylmer Fisher. *The design of experiments*. Oliver & Boyd, Edinburgh & London, 1937. ISBN 9780028446905.

[42] Marco Forgione, Dario Piga, and Alberto Bemporad. Efficient calibration of embedded mpc. *preprint arXiv:1911.13021*, 2019. doi: https://doi.org/10.48550/arxiv.1911.13021.

[43] Simone Formentin, Klaske Van Heusden, and Alireza Karimi. A comparison of model-based and data-driven controller tuning. *International Journal of Adaptive Control and Signal Processing*, 28(10):882–897, 2014. doi: https://doi.org/10.1002/acs.2415.

[44] Bengt Fornberg and Natasha Flyer. *A primer on radial basis functions with applications to the geosciences*. SIAM, 2015. ISBN 9781611974027.

[45] Peter I. Frazier. A tutorial on bayesian optimization. *arXiv preprint arXiv:1807.02811*, 2018. doi: https://doi.org/10.48550/arXiv.1807.02811.

[46] Johannes Fürnkranz and Eyke Hüllermeier. *Preference learning and ranking by pairwise comparison*. Springer, 2010. ISBN 9783642141249.

[47] Johannes Fürnkranz, Eyke Hüllermeier, Weiwei Cheng, and Sang-Hyeun Park. Preference-based reinforcement learning: a formal framework and a policy iteration algorithm. *Machine learning*, 89(1):123–156, 2012. doi: https://doi.org/10.1007/s10994-012-5313-8.

[48] Eduardo C. Garrido-Merchán and Daniel Hernández-Lobato. Dealing with categorical and integer-valued variables in bayesian optimization with gaussian processes. *Neurocomputing*, 380:20–35, 2020. doi: https://doi.org/10.1016/j.neucom.2019.11.004.

[49] Michael A. Gelbart, Jasper Snoek, and Ryan P. Adams. Bayesian optimization with unknown constraints. *arXiv preprint arXiv:1403.5607*, 2014. doi: https://doi.org/10.48550/arXiv.1403.5607.

[50] Giorgio Giorgi. A guided tour in constraint qualifications for nonlinear programming under differentiability assumptions. Technical report, University of Pavia, Department of Economics and Management, 2018.

[51] Gene H. Golub and Charles F. Van Loan. *Matrix computations*. JHU press, 2013. ISBN 9781421407944.

[52] Javier González, Zhenwen Dai, Andreas Damianou, and Neil D. Lawrence. Preferential bayesian optimization. In *International Conference on Machine Learning*, pages 1282–1291, 2017. URL `https://assets.amazon.science/da/1e/24ad7c354431bd0c2f64a6049269/preferential-bayesian-optimization.pdf`.

[53] Robert B. Gramacy and Herbert K. H. Lee. Cases for the nugget in modeling computer experiments. *Statistics and Computing*, 22(3):713–722, 2012. doi: https://doi.org/10.1007/s11222-010-9224-x.

[54] H.-M. Gutmann. A radial basis function method for global optimization. *Journal of global optimization*, 19(3):201–227, 2001. doi: https://doi.org/10.1023/A:1011255519438.

[55] David Halliday, Robert Resnick, and Jearl Walker. *Fundamentals of physics*. John Wiley & Sons, 2013. ISBN 9781118230718.

[56] Jiawei Han, Jian Pei, and Micheline Kamber. *Data mining: concepts and techniques*. Elsevier, 2011. ISBN 9789380931913.

[57] Per Christian Hansen, Victor Pereyra, and Godela Scherer. *Least squares data fitting with applications*. JHU Press, 2013. ISBN 9781421407869.

[58] Trevor Hastie, Robert Tibshirani, , and Jerome H. Friedman. *The elements of statistical learning: data mining, inference, and prediction*, volume 2. Springer, 2009. ISBN 9780387848570.

[59] Reiner Horst and Panos M. Pardalos. *Handbook of global optimization*, volume 2. Springer Science & Business Media, 2013. ISBN 9781461358381.

[60] Neil Houlsby, Ferenc Huszar, Zoubin Ghahramani, and Jose Hernández-lobato. Collaborative gaussian processes for preference learning. In *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012. URL `https://proceedings.neurips.cc/paper/2012/file/afdec7005cc9f14302cd0474fd0f3c96-Paper.pdf`.

[61] Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An introduction to statistical learning*, volume 112. Springer, 2013. ISBN 9781461471370.

[62] Momin Jamil and Xin-She Yang. A literature survey of benchmark functions for global optimisation problems. *International Journal of Mathematical Modelling and Numerical Optimisation*, 4(2):150–194, 2013. doi: http://doi.org/10.1504/IJMMNO.2013.055204.

[63] J. Jeswiet, M. Geiger, U. Engel, M. Kleiner, M. Schikorra, Joost Duflou, R. Neugebauer, P. Bariani, and S. Bruschi. Metal forming progress since 2000. *CIRP Journal of manufacturing Science and technology*, 1(1):2–17, 2008. doi: https://doi.org/10.1016/j.cirpj.2008.06.005.

[64] Mark E. Johnson, Leslie M. Moore, and Donald Ylvisaker. Minimax and maximin distance designs. *Journal of statistical planning and inference*, 26(2):131–148, 1990. doi: https://doi.org/10.1016/0378-3758(90)90122-B.

[65] Donald R. Jones. A taxonomy of global optimization methods based on response surfaces. *Journal of global optimization*, 21(4):345–383, 2001. doi: https://doi.org/10.1023/A:1012771025575.

[66] Donald R. Jones and Joaquim R. R. A. Martins. The direct algorithm: 25 years later. *Journal of Global Optimization*, 79(3):521–566, 2021. doi: https://doi.org/10.1007/s10898-020-00952-6.

[67] Donald R. Jones, Cary D. Perttunen, and Bruce E. Stuckman. Lipschitzian optimization without the lipschitz constant. *Journal of optimization Theory and Applications*, 79(1):157–181, 1993. doi: https://doi.org/10.1007/BF00941892.

[68] Donald R. Jones, Matthias Schonlau, and William J. Welch. Efficient global optimization of expensive black-box functions. *Journal of Global optimization*, 13(4):455–492, 1998. doi: https://doi.org/10.1023/A:1008306431147.

[69] V. Roshan Joseph and Lulu Kang. Regression-based inverse distance weighting with applications to computer experiments. *Technometrics*, 53(3):254–265, 2011. doi: https://doi.org/10.1198/TECH.2011.09154.

[70] S. Kazemzadeh Azad, O. Hasançebi, and O. K. Erol. Evaluating efficiency of big-bang big-crunch algorithm in benchmark engineering optimization problems. *Iran University of Science & Technology*, 1(3):495–505, 2011.

[71] John L. Kelley. *General topology*. Courier Dover Publications, 2017. ISBN 9780486815442.

[72] James Kennedy and Russell Eberhart. Particle swarm optimization. *Proceedings of ICNN'95-international conference on neural networks*, 4:1942–1948, 1995. doi: https://doi.org/10.1109/ICNN.1995.488968.

[73] Dohyung Kim and Hyun-Shik Oh. Black-box optimization of pid controllers for aircraft maneuvering control. *International Journal of Control, Automation and Systems*, 20(3):703–714, 2022. doi: https://doi.org/10.1007/s12555-020-0915-6.

[74] Gary King and Langche Zeng. Logistic regression in rare events data. *Political analysis*, 9(2):137–163, 2001. doi: https://doi.org/10.1093/oxfordjournals.pan.a004868.

[75] Scott Kirkpatrick, C. Daniel Gelatt Jr., and Mario P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983. doi: https://doi.org/10.1126/science.220.4598.671.

[76] Chun-Wa Ko, Jon Lee, and Maurice Queyranne. An exact algorithm for maximum entropy sampling. *Operations Research*, 43(4):684–691, 1995. doi: https://doi.org/10.1287/opre.43.4.684.

[77] John Kruschke. *Doing Bayesian data analysis: A tutorial with R, JAGS, and Stan*. Academic Press, 2014. ISBN 9780124058880.

[78] Abhishek Kumar, Guohua Wu, Mostafa Z. Ali, Rammohan Mallipeddi, Ponnuthurai Nagaratnam Suganthan, and Swagatam Das. A test-suite of non-convex constrained optimization problems from the real-world and some baseline results. *Swarm and Evolutionary Computation*, 56:100693, 2020. doi: https://doi.org/10.1016/j.swevo.2020.100693.

[79] Harold J. Kushner. A new method of locating the maximum point of an arbitrary multipeak curve in the presence of noise. *Journal of Basic Engineering*, 1964. doi: https://doi.org/10.1115/1.3653121.

[80] Hoai A. Le Thi, A. Ismael F. Vaz, and Luis Nunes Vicente. Optimizing radial basis functions by dc programming and its use in direct search for global derivative-free optimization. *Top*, 20 (1):190–214, 2012. doi: https://doi.org/10.1007/s11750-011-0193-9.

[81] Martin J. Lee and Ken Kang Too Tsang. *Nonlinear Algebra in an ACORN: With Applications to Deep Learning*. World Scientific, 2018. ISBN 9789813271517.

[82] Lei Li, Haihong Huang, Fu Zhao, John W. Sutherland, and Zhifeng Liu. An energy-saving method by balancing the load of operations for hydraulic press. *IEEE/ASME Transactions on Mechatronics*, 22(6):2673–2683, 2017. doi: https://doi.org/10.1109/TMECH.2017.2759228.

[83] Hsuan-Tien Lin, Chih-Jen Lin, and Ruby C. Weng. A note on platt's probabilistic outputs for support vector machines. *Machine learning*, 68(3):267–276, 2007. doi: https://doi.org/10.1007/s10994-007-5018-6.

[84] G. P. Liu, J. B. Yang, and J. F. Whidborn. *Multiobjective optimisation and control*. Research Studies Press Ltd, 2002. ISBN 9780863802645.

[85] Daniel James Lizotte. *Practical bayesian optimization*. PhD thesis, University of Alberta, 2008.

[86] Lennart Ljung. *System identification*. Springer, 1998. ISBN 9780136566953.

[87] Marco Locatelli and Fabio Schoen. *Global optimization: theory, algorithms, and applications*. SIAM, 2013. ISBN 9781611972665.

[88] Noah D. Manring and Roger C. Fales. *Hydraulic control systems*. John Wiley & Sons, 2019. ISBN 9781119416470.

[89] Alonso Marco, Philipp Hennig, Jeannette Bohg, Stefan Schaal, and Sebastian Trimpe. Automatic lqr tuning based on gaussian process global optimization. *Proceedings of the 2016 IEEE International Conference on Robotics and Automation (ICRA)*, 2016. doi: https://doi.org/10.1109/ICRA.2016.7487144.

[90] Rafael Martí, Jose A. Lozano, Alexander Mendiburu, and Leticia Hernando. Multi-start methods. *Handbook of heuristics*, pages 155–175, 2018. doi: https://doi.org/10.1007/0-306-48056-5_12.

[91] Michael D. McKay, Richard J. Beckman, and William J. Conover. A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*, 42(1):55–61, 2000. doi: http://doi.org/10.1080/00401706.2000.10485979.

[92] Sudhanshu K. Mishra. Some new test functions for global optimization and performance of repulsive particle swarm method. *Available at SSRN 926132*, 2006. URL `https://papers.ssrn.com/sol3/papers.cfm?abstract_id=926132`.

[93] Jonas Mockus. *Bayesian approach to global optimization: theory and applications*, volume 37. Springer Science & Business Media, 1989. ISBN 9780792301158.

[94] Jonas Mockus, Vytautas Tiesis, and Antanas Zilinskas. The application of bayesian methods for seeking the extremum. *Towards global optimization*, 2(117-129):2, 1978.

[95] Max D. Morris and Toby J. Mitchell. Exploratory designs for computational experiments. *Journal of statistical planning and inference*, 43(3):381–402, 1995. doi: https://doi.org/10.1016/0378-3758(94)00035-T.

[96] Kwara Nantomah. On some properties of the sigmoid function. *Asia Mathematika*, 2019.

[97] Arnold Neumaier. Complete search in continuous global optimization and constraint satisfaction. *Acta numerica*, 13:271–369, 2004. doi: https://doi.org/10.1017/S0962492904000194.

[98] Matthias Neumann-Brosig, Alonso Marco, Dieter Schwarzmann, and Sebastian Trimpe. Data-efficient autotuning with bayesian optimization: An industrial control study. *IEEE Transactions on Control Systems Technology*, 28(3):730–740, 2020. doi: https://doi.org/10.1109/TCST.2018.2886159.

[99] Hiroki Nishimura and Efe A. Ok. Preference structures. Technical report, mimeo, NYU, 2018.

[100] Jorge Nocedal and Stephen J. Wright. *Numerical optimization*. Springer, 1999. ISBN 0387987932.

[101] Aidan O'dwyer. *Handbook of PI and PID controller tuning rules*. Imperial College Press, 2009. ISBN 9781848162426.

[102] Katsuhiko Ogata. *Modern control engineering*, volume 5. Prentice hall Upper Saddle River, NJ, 2010. ISBN 9780136156734.

[103] Efe A. Ok. Utility representation of an incomplete preference relation. *Journal of Economic Theory*, 104(2):429–449, 2002. doi: https://doi.org/10.1006/jeth.2001.2814.

[104] Efe A. Ok. *Real analysis with economic applications*. Princeton University Press, 2011. ISBN 9780691117683.

[105] K. Osakada, Keisuke Mori, T. Altan, and P. Groche. Mechanical servo press technology for metal forming. *CIRP annals*, 60(2):651–672, 2011. doi: https://doi.org/10.1016/j.cirp.2011.05.007.

[106] John Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in large margin classifiers*, 10(3):61–74, 1999.

[107] Michael J. D. Powell. The theory of radial basis function approximation in 1990. *Advances in numerical analysis*, pages 105–210, 1992.

[108] Davide Previtali, Mirko Mazzoleni, Antonio Ferramosca, and Fabio Previdi. A unified surrogate-based scheme for black-box and preference-based optimization. *arXiv preprint arXiv.2202.01468*, 2022. doi: https://doi.org/10.48550/arXiv.2202.01468.

[109] Davide Previtali, Mirko Mazzoleni, Antonio Ferramosca, and Fabio Previdi. Glisp-r: A preference-based optimization algorithm with convergence guarantees. *Computational Optimization and Applications*, pages 1–38, 2023. doi: https://doi.org/10.1007/s10589-023-00491-2.

[110] James Blake Rawlings, David Q. Mayne, and Moritz Diehl. *Model predictive control: theory, computation, and design*, volume 2. Nob Hill Publishing Madison, WI, 2017. ISBN 9780975937754.

[111] Rommel G. Regis. Stochastic radial basis function algorithms for large-scale optimization involving expensive black-box objective and constraint functions. *Computers and Operations Research*, 38(5):837–853, 2011. ISSN 0305-0548. doi: https://doi.org/10.1016/j.cor.2010.09.013.

[112] Rommel G Regis. Constrained optimization by radial basis function interpolation for high-dimensional expensive black-box problems with infeasible initial points. *Engineering Optimization*, 46(2):218–243, 2014. doi: https://doi.org/10.1080/0305215X.2013.765000.

[113] Rommel G. Regis. Multi-objective constrained black-box optimization using radial basis function surrogates. *Journal of Computational Science*, 16:140–155, 2016. ISSN 1877-7503. doi: https://doi.org/10.1016/j.jocs.2016.05.013.

[114] Rommel G. Regis. A survey of surrogate approaches for expensive constrained black-box optimization. In *World Congress on Global Optimization*, pages 37–47. Springer, 2019. doi: https://doi.org/10.1007/978-3-030-21803-4_4.

[115] Rommel G. Regis and Christine A. Shoemaker. Constrained global optimization of expensive black box functions using radial basis functions. *Journal of Global optimization*, 31(1):153–171, 2005. doi: https://doi.org/10.1007/s10898-004-0570-0.

[116] Rommel G. Regis and Christine A. Shoemaker. A stochastic radial basis function method for the global optimization of expensive functions. *INFORMS Journal on Computing*, 19(4):497–509, 2007. doi: https://doi.org/10.1287/ijoc.1060.0182.

[117] Rommel G. Regis and Stefan M. Wild. Conorbit: constrained optimization by radial basis function interpolation in trust regions. *Optimization Methods and Software*, 32(3):552–580, 2017. doi: https://doi.org/10.1080/10556788.2016.1226305.

[118] Luis Miguel Rios and Nikolaos V. Sahinidis. Derivative-free optimization: a review of algorithms and comparison of software implementations. *Journal of Global Optimization*, 56(3): 1247–1293, 2013. doi: https://doi.org/10.1007/s10898-012-9951-y.

[119] Shmuel Rippa. An algorithm for selecting a good value for the parameter c in radial basis function interpolation. *Advances in Computational Mathematics*, 11(2):193–210, 1999. doi: https://doi.org/10.1023/A:1018975909870.

[120] Loris Roveda, Beatrice Maggioni, Elia Marescotti, Asad Ali Shahid, Andrea Maria Zanchettin, Alberto Bemporad, and Dario Piga. Pairwise preferences-based optimization of a path-based velocity planner in robotic sealing tasks. *IEEE Robotics and Automation Letters*, 6(4):6632–6639, 2021. doi: https://doi.org/10.1109/LRA.2021.3094479.

[121] Walter Rudin. *Principles of mathematical analysis*, volume 3. McGraw-hill New York, 1976. ISBN 9780070856134.

[122] Daniel J. Russo, Benjamin Van Roy, Abbas Kazerouni, Ian Osband, and Zheng Wen. A tutorial on thompson sampling. *Foundations and Trends® in Machine Learning*, 11(1):1–96, 2018. doi: http://doi.org/10.1561/9781680834710.

[123] Jerome Sacks, William J. Welch, Toby J. Mitchell, and Henry P. Wynn. Design and analysis of computer experiments. *Statistical science*, 4(4):409–423, 1989. doi: http://doi.org/10.1214/ss/1177012413.

[124] Thomas J. Santner, Brian J. Williams, William I. Notz, and Brain J. Williams. *The design and analysis of computer experiments*. Springer, 2019. ISBN 9781493988471.

[125] Michael James Sasena. *Flexibility and efficiency enhancements for constrained global design optimization with kriging approximations*. PhD thesis, University of Michigan, 2002.

[126] Robert Schaback. Error estimates and condition numbers for radial basis function interpolation. *Advances in Computational Mathematics*, 3(3):251–264, 1995. doi: https://doi.org/10.1007/BF02432002.

[127] Eric Schechter. *Handbook of Analysis and its Foundations*. Academic Press, 1996. ISBN 9780126227604.

[128] Matthias Schonlau. *Computer experiments and global optimization*. PhD thesis, University of Waterloo, 1997.

[129] Jssai Schur. Bemerkungen zur theorie der beschränkten bilinearformen mit unendlich vielen veränderlichen. 1911.

[130] Sergei Shabanov. *Concepts in Calculus III, Multivariable Calculus*. University Press of Florida, 2012. ISBN 9781616101626.

[131] Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P. Adams, and Nando De Freitas. Taking the human out of the loop: A review of bayesian optimization. *Proceedings of the IEEE*, 104 (1):148–175, 2015. doi: https://doi.org/10.1109/JPROC.2015.2494218.

[132] Donald Shepard. A two-dimensional interpolation function for irregularly-spaced data. In *Proceedings of the 1968 23rd ACM national conference*, pages 517–524, 1968. doi: https://doi.org/10.1145/800186.810616.

[133] Željko Šitum. Force and positon control of a hydraulic press. *Ventil*, 17(4):314–320, 2011.

[134] Jasper Snoek, Hugo Larochelle, and Ryan P. Adams. Practical bayesian optimization of machine learning algorithms. In *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012. URL `https://proceedings.neurips.cc/paper/2012/file/05311655a15b75fab86956663e1819cd-Paper.pdf`.

[135] András Sobester, Alexander Forrester, and Andy Keane. *Engineering design via surrogate modelling: a practical guide*. John Wiley & Sons, 2008. ISBN 9780470060681.

[136] Francisco J. Solis and Roger J.-B. Wets. Minimization by random search techniques. *Mathematics of operations research*, 6(1):19–30, 1981. doi: https://doi.org/10.1287/moor.6.1.19.

[137] Niranjan Srinivas, Andreas Krause, Sham M. Kakade, and Matthias Seeger. Gaussian process optimization in the bandit setting: No regret and experimental design. *arXiv preprint arXiv:0912.3995*, 2009. doi: https://doi.org/10.48550/arXiv.0912.3995.

[138] Erwin Stinstra, Dick den Hertog, Peter Stehouwer, and Arjen Vestjens. Constrained maximin designs for computer experiments. *Technometrics*, 45(4):340–346, 2003. doi: https://doi.org/10.1198/004017003000000168.

[139] Jörg Stork, Agoston E. Eiben, and Thomas Bartz-Beielstein. A new taxonomy of global optimization algorithms. *Natural Computing*, pages 1–24, 2020. doi: https://doi.org/10.1007/s11047-020-09820-4.

[140] Chungeng Sun, Jinhui Fang, Jianhua Wei, and Bo Hu. Nonlinear motion control of a hydraulic press based on an extended disturbance observer. *IEEE Access*, 6:18502–18510, 2018. doi: https://doi.org/10.1109/ACCESS.2018.2813317.

[141] Gerald Tesauro. Connectionist learning of expert preferences by comparison training. In *Advances in Neural Information Processing Systems*, volume 1. Morgan-Kaufmann, 1988. URL `https://proceedings.neurips.cc/paper/1988/file/a8baa56554f96369ab93e4f3bb068c22-Paper.pdf`.

[142] P. B. Thanedar, J. S. Arora, G. Y. Li, and T. C. Lin. Robustness, generality and efficiency of optimization algorithms for practical applications. *Structural optimization*, 2(4):203–212, 1990. doi: https://doi.org/10.1007/BF01748225.

[143] Aimo Torn and Antanas Zilinskas. *Global Optimization*. Lecture Notes in Computer Science, 1989. ISBN 9783540508717.

[144] Frans Van Den Bergh. *An analysis of particle swarm optimizers*. PhD thesis, University of Pretoria, 2007.

[145] Paul M. J. Van den Hof. *System identification - Data-driven modelling of dynamic systems*. Lecture notes, 2012.

[146] A. Ismael F. Vaz and Luis Nunes Vicente. A particle swarm pattern search method for bound constrained global optimization. *Journal of Global Optimization*, 39(2):197–219, 2007. doi: https://doi.org/10.1007/s10898-007-9133-5.

[147] A. Ismael F. Vaz and Luis Nunes Vicente. Pswarm: a hybrid solver for linearly constrained global derivative-free optimization. *Optimization Methods & Software*, 24(4-5):669–685, 2009. doi: https://doi.org/10.1080/10556780902909948.

[148] Francesco Vivarelli. *Studies on the generalisation of Gaussian processes and Bayesian neural networks*. PhD thesis, Aston University, 1998.

[149] Ky Khac Vu, Claudia d'Ambrosio, Youssef Hamadi, and Leo Liberti. Surrogate-based methods for black-box optimization. *International Transactions in Operational Research*, 24(3):393–424, 2017. doi: https://doi.org/10.1111/itor.12292.

[150] Steven Xiaogang Wang. *Maximum weighted likelihood estimation*. PhD thesis, University of British Columbia, 2001.

[151] Yilun Wang and Christine A. Shoemaker. A general stochastic algorithmic framework for minimizing expensive black box objective functions based on surrogate models and sensitivity analysis. *arXiv preprint arXiv:1410.6271*, 2014. doi: https://doi.org/10.48550/arXiv.1410.6271.

[152] Stefan M. Wild, Rommel G. Regis, and Christine A. Shoemaker. Orbit: Optimization by radial basis function interpolation in trust-regions. *SIAM Journal on Scientific Computing*, 30(6): 3197–3219, 2008. doi: https://doi.org/10.1137/070691814.

[153] Christopher K. Williams and Carl Edward Rasmussen. *Gaussian processes for machine learning*. MIT press Cambridge, MA, 2005. ISBN 9780262182539.

[154] Dawei Zhan and Huanlai Xing. Expected improvement for expensive optimization: a review. *Journal of Global Optimization*, 78(3):507–544, 2020. doi: https://doi.org/10.1007/s10898-020-00923-x.

[155] Mengjia Zhu, Alberto Bemporad, and Dario Piga. Preference-based mpc calibration. *arXiv preprint arXiv:2003.11294*, 2020. doi: https://doi.org/10.48550/arXiv.2003.11294.

[156] Mengjia Zhu, Dario Piga, and Alberto Bemporad. C-glisp: Preference-based global optimization under unknown constraints with applications to controller calibration. *IEEE Transactions on Control Systems Technology*, 2021. doi: http://doi.org/10.1109/TCST.2021.3136711.