Advanced Geometrical Test for Interval Branch and Bound methods

Mihály Gencsi and Boglárka G.-Tóth

Department of Computational Optimization, University of Szeged, Árpád square 2., Szeged, H-6720, Hungary.

Contributing authors: gencsi@inf.szte.hu; boglarka@inf.szte.hu;

Abstract

The Interval Branch and Bound (IBB) method is a widely used approach for solving nonlinear programming problems, especially when a rigorous solution is required. It uses Interval Arithmetic to handle rounding errors. Although numerous variants of the IBB method have been proposed in the literature, relatively few implementations incorporate Karush-Kuhn-Tucker or Fritz-John (FJ) optimality conditions to eliminate nonoptimal boxes. Using the FJ conditions involves solving an interval-valued linear system of equations. However, the solution is often overestimated, making the test ineffective.

This study is based on the geometrical interpretation of the optimality condition and involves the development of an Advanced Geometrical Test to be performed before solving the optimality conditions. The theoretical validity of the new techniques has been established, and the efficiency of the new test was evaluated through computational experiments on a benchmark set of 374 test cases. Nine variants of the IBB algorithm were compared, differing only in the tests used to check the optimality conditions. The IBB method combined with the Advanced Geometrical Test and the Lagrange estimator method achieves the best overall performance, earning a 2.1 times speed-up.

Keywords: Global Optimization, Interval Arithmetic, Branch and Bound method, Optimality conditions, Geometrical Test

1 Introduction

Many applications require rigorous solutions to mathematical problems, such as finding stability points in physics and chemistry. We focus on solving inequality-constrained global optimization problems, specifically the following n-dimensional nonlinear problem:

$$\begin{aligned} \underset{x \in \boldsymbol{y} \subseteq \mathbb{R}^n}{\text{minimize}} \quad & f(x) \\ \text{subject to} \quad & g_i(x) \leq 0, \quad i \in M_c, \end{aligned} \tag{1}$$

where $f, g_i : \mathbb{R}^n \to \mathbb{R}, i \in M_c$ are continuously differentiable nonlinear functions and $\boldsymbol{y} = [\underline{y}, \overline{y}]$ defines the bound constraints. The bound constraints can be written for x_i as $p_{i_u}(x) = x_i - \overline{y_i} \leq 0$ and $p_{i_l}(x) = \underline{y_i} - x_i \leq 0$, where $\overline{y_i}$ and $\underline{y_i}$ are the upper and lower bounds of x_i, respectively. Thus, the bound constraints are $p_j(x) \leq 0, j \in M_b = \{i_u, i_l \mid i \in N\}$, where $N = \{1, \ldots, n\}$.

Solving (1) is challenging due to many local minima. Traditional nonlinear optimization methods often only find local optima and lack the ability to prove global optimality. The Interval Branch and Bound (IBB) method is one of the most effective approaches, integrating the Branch and Bound framework with Interval Arithmetic for bound calculations.

Various IBB variants exist [1–7]. However, in practice, many do not use optimality conditions to eliminate nonoptimal subproblems in the constrained case. The Fritz-John (FJ) optimality conditions generalize the Karush-Kuhn-Tucker (KKT) conditions without requiring regularity assumptions [8]. Using the optimality conditions means to solve an interval-valued linear system of equations. Common interval solvers include the iterative Interval Gauss-Seidel method (IGS) [9], the direct Interval LU decomposition (ILUD) method [10], and the Hull method [11], which is efficient for regular matrices. Preconditioning, such as the use of midpoint preconditioners [12], can improve solvability. However, solving the Fritz-John conditions in many cases is impractical, requiring high computational effort and resulting in an overestimated inclusion as the solution.

Various approaches are introduced to solve the FJ conditions [11, 13], including bounding Lagrange multipliers, enclosing optimal points, and using Taylor expansions. Some studies [14] consider the FJ conditions as contractors. However, no computational results or usable implementations are documented. In [15], the authors compared different FJ solution methods within IBB. Furthermore, in [16], the geometrical interpretation of the optimality conditions is presented, and a test is introduced before solving the FJ conditions.

This work makes the following contributions. It collects the possible uses of the geometrical interpretation of the optimality conditions. We present the theoretical results and the Advanced Geometrical Test within the IBB framework. Specifically, the test can either reject a given subproblem if it cannot contain any optimal solution, shrink the subproblem, or decide if the optimality conditions may or may not work. It works as a preliminary test before the optimality conditions and speeds up the method considerably.

The structure of the paper is organized as follows. Section 2 introduces the fundamental concepts of Interval Arithmetic (IA), the designed Interval Branch and Bound (IBB) method, including the interval optimality conditions and the corresponding solution methods. In Section 2.4, the geometrical interpretation of the Fritz-John optimality conditions is extended to intervals. Section 3 outlines our theoretical results. Based on our findings, Section 4 describes the Advanced Geometrical Test. The computational results are reported in Section 5, based on the benchmarks used to compare the different variants of the IBB algorithm. Finally, Section 6 concludes the paper with a summary of our findings and suggestions for future research directions.

2 Preliminaries

In this section, we introduce the basic notation of Interval Arithmetic (IA), the Fritz-John optimality conditions, and the improved IBB method.

2.1 Interval Arithmetic

In Interval Arithmetic, the main idea is to replace numbers with intervals to eliminate rounding errors and measurement inaccuracies, and to compute bounds on function values.

An *interval* is represented by $\boldsymbol{x} = [\underline{x}, \overline{x}]$ with bounds $\underline{x}$ and $\overline{x}$. The concept extends to an n-dimensional interval vector or *intervalbox*, $\boldsymbol{x} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n) \in \mathbb{I}^n = \mathbb{I} \times \cdots \times \mathbb{I}$, where $\mathbb{I}$ is the interval set. Notable attributes include the *midpoint*, $\operatorname{mid}(\boldsymbol{x}) = \frac{1}{2}(\overline{x} + \underline{x})$, and the *width*, $\operatorname{wid}(\boldsymbol{x}) = \overline{x} - \underline{x}$, extending to boxes as $\operatorname{mid}(\boldsymbol{x}) = (\operatorname{mid}(\boldsymbol{x}_1), \ldots, \operatorname{mid}(\boldsymbol{x}_n))^T$ and $\operatorname{wid}(\boldsymbol{x}) = \max_{i \in N} \operatorname{wid}(\boldsymbol{x}_i)$. The *interval hull*, denoted by the operator $\bigsqcup$, is defined as the smallest box that contains the union of the given boxes. For example, consider two boxes $\boldsymbol{x} = ([2,4],[1,3])$ and $\boldsymbol{y} = ([5,6],[2,5])$. Their interval hull is the smallest enclosing box, calculated by taking the minimum of the lower bounds and the maximum of the upper bounds, $([\min(2,5), \max(4,6)], [\min(1,2), \max(3,5)]) = ([2,6],[1,5])$.

Arithmetic operations $(+, -, \cdot, /)$ naturally extend to intervals, with $\boldsymbol{x} \odot \boldsymbol{y} = \{x \odot y : x \in \boldsymbol{x}, y \in \boldsymbol{y}\}$ for $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{I}$, where $\odot$ is any operator, with division restricted to $0 \notin \boldsymbol{y}$. However, we also use the extended interval division that handles division by zero [17]. Elementary functions such as sin, cos, and exp are extended to the intervals [7, 11, 18]. An *inclusion function* $\mathbf{f} : \mathbb{I}^n \to \mathbb{I}$ ensures that the range of the function $f : \mathbb{R}^n \to \mathbb{R}$ for any intervalbox $\boldsymbol{x}$ is within $\mathbf{f}(\boldsymbol{x}) = [\underline{\mathbf{f}}(\boldsymbol{x}), \overline{\mathbf{f}}(\boldsymbol{x})]$, i.e., $f(\boldsymbol{x}) \subseteq \mathbf{f}(\boldsymbol{x})$. Although these functions are inclusive, they may overestimate ranges. The upper and lower bounds of f over an interval $\boldsymbol{x}$ are naturally obtained by $\underline{\mathbf{f}}(\boldsymbol{x})$ and $\overline{\mathbf{f}}(\boldsymbol{x})$.

The simplest inclusion function is the *natural interval extension*, in which variables and real functions are replaced with their interval counterparts. The quadratically convergent, but computationally more expensive inclusion is the *centred form*, $\mathbf{f}_c(\boldsymbol{x}) = \mathbf{f}(c) + \boldsymbol{\nabla}\mathbf{f}(\boldsymbol{x}) \cdot (\boldsymbol{x} - c)$, where $c \in \boldsymbol{x}$ is typically the centre of the box and $\boldsymbol{\nabla}\mathbf{f}(\boldsymbol{x}) = \left(\frac{\partial \mathbf{f}(\boldsymbol{x})}{\partial \boldsymbol{x}_1}, \ldots, \frac{\partial \mathbf{f}(\boldsymbol{x})}{\partial \boldsymbol{x}_n}\right)$ represents the gradient enclosure of f over the box $\boldsymbol{x}$. Automatic Differentiation [19] effectively calculates gradients. For a deeper insight into Interval Arithmetic, see [11, 14].

2.2 Fritz-John Optimality Conditions

The Fritz-John (FJ) conditions provide the necessary optimality conditions in nonlinear programming. For problem (1), the FJ conditions [8] for a locally optimal point x are the system of equations

$$L(x,\mu) = \mu_0 \nabla f(x) + \sum_{i \in M_b} \mu_i \nabla p_i(x) + \sum_{j \in M_c} \mu_j \nabla g_j(x) = 0 \tag{2}$$

$$\mu_i p_i(x) = 0, \quad i \in M_b \tag{3}$$

$$\mu_j g_j(x) = 0, \quad j \in M_c \tag{4}$$

$$\mu_i \geq 0, \quad i \in M_b \cup M_c \cup \{0\} \tag{5}$$

$$\mu \neq 0, \tag{6}$$

where μ_i are the Lagrange multipliers, M_b and M_c are the set of bound and general constraints, respectively. The FJ conditions include the multiplier μ_0, while in the Karush-Kuhn-Tucker (KKT) conditions, μ_0 is fixed at 1, although it requires additional regularity conditions [20].

In the interval version of the Fritz-John optimality conditions, all functions are replaced by their inclusion functions, and all real vectors as μ, x are replaced by interval vectors $\boldsymbol{\mu}, \boldsymbol{x}$. The Normalized Interval Fritz-John Optimality Condition System (NFJ) [15] for a given box $\boldsymbol{x}$ is the following interval-valued linear system of equations, written in matrix form:

$$\boldsymbol{\phi}(\boldsymbol{t}) = \begin{bmatrix} \boldsymbol{\mu}_0 + \sum_{i \in B \cup C} \boldsymbol{\mu}_i - 1 \\ \boldsymbol{\mu}_0 \cdot \boldsymbol{\nabla}\mathbf{f}(\boldsymbol{x}) + \sum_{i \in B} \boldsymbol{\mu}_i \cdot \boldsymbol{\nabla}\mathbf{p}_i(\boldsymbol{x}) + \sum_{j \in C} \boldsymbol{\mu}_j \cdot \boldsymbol{\nabla}\mathbf{g}_j(\boldsymbol{x}) \\ \boldsymbol{\mu}_i \cdot \mathbf{p}_i(\boldsymbol{x}) \qquad i \in B \\ \boldsymbol{\mu}_j \cdot \mathbf{g}_j(\boldsymbol{x}) \qquad j \in C \end{bmatrix} = 0, \tag{7}$$

where $\mathbf{f}(\boldsymbol{x})$, $\mathbf{p}_i(\boldsymbol{x})$, $\mathbf{g}_j(\boldsymbol{x})$ are the inclusion functions, $\boldsymbol{\nabla}\mathbf{f}(\boldsymbol{x})$, $\boldsymbol{\nabla}\mathbf{p}_i(\boldsymbol{x})$, $\boldsymbol{\nabla}\mathbf{g}_j(\boldsymbol{x})$ are the inclusions of the gradients for $f(x), p_i(x), g_j(x)$, respectively. The system is formalized only in active constraints, i.e. $B = \{i \in M_B \mid 0 \in \mathbf{p}_i(\boldsymbol{x})\}$, $C = \{i \in M_C \mid 0 \in \mathbf{g}_j(\boldsymbol{x})\}$, and the vector of variables is $\boldsymbol{t} = [\boldsymbol{x}, \boldsymbol{\mu}]^T$, which is $n + |B| + |C| + 1$ dimensional. Apart from the first constraint, normalization also means that the Lagrange multipliers are $\boldsymbol{\mu}_i \subseteq [0,1]$ instead of $[0,\infty)$. A solution of this interval-valued system of equations means that there is a real coefficient matrix contained in the interval matrix for which the system holds. In this sense, $\boldsymbol{\phi}(\boldsymbol{t}) = 0$ is solvable if $0 \in \boldsymbol{\phi}(\boldsymbol{t})$. The interested reader can find more information about the interval FJ conditions [6, 15, 16].

In [15], the best methods found to solve (7) are the Lagrange estimator (Lag) method and the Lagrange estimator + NFJ (Lag+NFJ) method. The Lag method computes bounds for the Lagrange multipliers to verify the current box. If the resulting bounds are negative or inconsistent with the system of equations, the box is discarded. The Lag+NFJ method extends the Lag method by incorporating a Newton step. If the box is not discarded during multiplier verification, the Newton step attempts to

contract or split the box further to remove non-optimal regions. Detailed descriptions of the two methods can be found in [15] or in Appendix A.

2.3 Interval Branch and Bound method

The Branch and Bound framework solves optimization problems by iteratively dividing the search space (branching) and pruning non-optimal regions based on bounds (bounding). The Interval Branch and Bound (IBB) method integrates Interval Arithmetic into this process using five core rules: selection, bounding, discarding, termination, and division. As this work focuses on the Advanced Geometrical Test, the general forms of the remaining rules are used as described in [16].

The algorithm is designed as follows. We are given the objective function f, the constraint functions $g_i, i \in M_c$, and the search region $\boldsymbol{y}$. The method uses two main lists: a work list for unexplored subproblems and a result list for potential solutions. These lists are sorted by the lower bound of the inclusion function. First, the global upper bound is set to infinity, and we evaluate the initial box using natural inclusion. Using the **Feasibility Test**, if any constraint is strictly violated, the box is infeasible. Constraints whose inclusion intervals contain zero are flagged as active, which classifies the box as undetermined. Conversely, boxes with no active constraints are considered feasible. If the initial box is infeasible, the algorithm terminates. Otherwise, the initial box is placed in the work list.

The main loop is repeated as long as the work list is non empty and the time limit permits. The **selection rule** returns the box with the smallest lower bound, to which the Basic discarding tests (excluding the Feasibility Test, see Section 2.3.1) and the Advanced Discarding Tests (Section 2.3.2) are applied. If the box is neither discarded nor narrowed, the **division rule** splits it into four subboxes based on the two widest dimensions. The resulting boxes are validated using the Basic tests, including the Feasibility Test. Finally, boxes satisfying the termination rule (based on width thresholds) are moved to the result list, while the remaining ones are added to the work list.

After the main loop, we use the **CutOff Test** to discard subboxes from the result list. This test discards any subproblem for which the computed lower bound is greater than the global upper bound. Finally, we return the result list once the work list is empty. However, when the algorithm reaches the maximum allowed runtime, we return all the remaining boxes, including the work list.

2.3.1 Basic discarding tests

In the Basic discarding tests, two **bounding rules** are used for the objective function: the natural inclusion function and the centred form $\mathbf{f}(c) + \boldsymbol{\nabla}\mathbf{f}(\boldsymbol{x})(\boldsymbol{x} - c)$, where $c = \text{mid}(\boldsymbol{x})$. Then, the Cutoff Test is applied. If the box is not discarded, the **Feasibility Test** (if enabled) removes infeasible boxes and updates the active constraints. This step is refined by using the centred form for constraints whenever gradient information has been computed. If the box centre is feasible, the global upper bound is updated, which triggers a secondary CutOff test to remove suboptimal nodes from the work list.

2.3.2 Advanced Discarding Tests

Some discarding tests use higher-order information to verify optimality conditions and discard nonoptimal boxes. These tests differ for strictly feasible and undetermined boxes.

In the Advanced Discarding Tests, the tests are organized so that those requiring the least additional information are executed first. In the strictly feasible case, the **Monotonicity Test** is performed first to check whether the gradient inclusion excludes zero. In that case, the function is monotonic over the box, which is then either discarded or reduced to its intersection with the boundary. Subsequent tests are applied only to non-deformed boxes, which are defined as those where the ratio of minimum to maximum diameter exceeds a specific parameter. This reduces unnecessary checks and ensures convergence. The **Non-convexity Test** analyzes the inclusion of the Hessian matrix. The presence of a negative diagonal element proves non-convexity. The box is discarded if it is strictly interior or reduced to the boundary faces. The **Newton Test** is then applied to interior boxes to narrow the search space around stationary points by solving an interval linear system using the Interval Gauss-Seidel method and extended interval division. These tests are described in more detail in [15] or in Appendix B.

In the undetermined case, when active constraints are present, the Fritz-John Test (see Section 2.2) is executed with or without the Advanced Geometrical Test (see Section 2.3.2). We use a preliminary test before the Fritz-John Test, which ensures that bound constraints are not active on both sides for any dimension and that the number of variables exceeds the number of active constraints. This is necessary for solving the interval linear system and helps avoid unnecessary computations in the optimality tests. It should be noted that in this version of our method, the preliminary test is only performed before the Fritz-John Test. This differs from the earlier approach presented in [15], which applied the preliminary test before both the Geometrical and the Fritz-John optimality tests.

2.4 Geometrical interpretation of optimality conditions

Many books and articles discuss the geometrical interpretation of optimality conditions. Figure 1a, taken from [16], illustrates the geometrical interpretation of the optimality conditions extended to intervals. The figure examines the optimality conditions at the interval $\boldsymbol{x}$. The feasible region X (gray area) is shown, as well as the inclusion of the gradients of the active constraints $\boldsymbol{\nabla}\mathbf{g}_1(\boldsymbol{x}), \boldsymbol{\nabla}\mathbf{g}_2(\boldsymbol{x})$, a possible feasible direction r, and the inclusion of the negative gradients of the objective $-\boldsymbol{\nabla}\mathbf{f}(\boldsymbol{x})$.

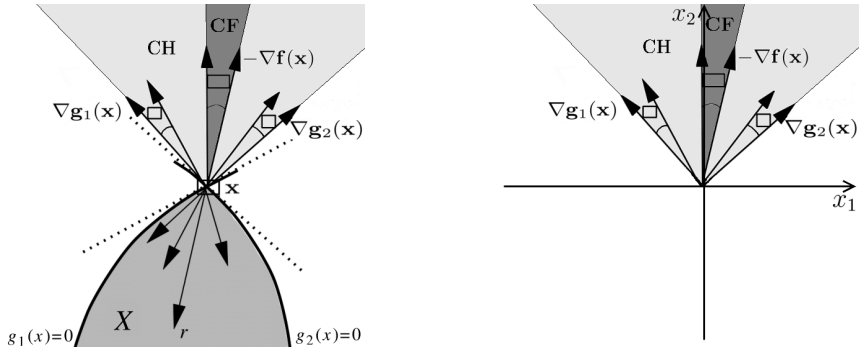
Let us define the negative cone of the objective's gradient,

$$\mathbf{CF} = \{d \in \mathbb{R}^n \mid d \in -\mu_0 \boldsymbol{\nabla}\mathbf{f}(\boldsymbol{x}), \mu_0 \geq 0\},$$

and the conic hull of the active constraints' gradients,

$$\mathbf{CH} = \left\{ d \in \mathbb{R}^n \;\middle|\; d \in \sum_{i \in B} \mu_i \boldsymbol{\nabla}\mathbf{p}_i(\boldsymbol{x}) + \sum_{i \in C} \mu_i \boldsymbol{\nabla}\mathbf{g}_i(\boldsymbol{x}), \mu \geq 0, \mu \neq 0 \right\}.$$

As shown in Figure 1a, the FJ necessary conditions require that $\mathbf{CF} \cap \mathbf{CH} \neq \emptyset$. In other words, there must exist a direction from the cone **CF**, which is also contained in the conic hull **CH**. Furthermore, $0 \in \mathbf{CH}$ implies that the regularity conditions may not be satisfied, since they are needed to allow $\mu_0 = 0$. Figure 1a illustrates **CH** (light gray), which contains **CF** (dark gray). Figures 1b show the gradient enclosures and the cones in the vector space for Figure 1a. It can be seen that here we consider the same condition, i.e. $\mathbf{CF} \cap \mathbf{CH} \neq \emptyset$.



(a) Geometrical meaning (b) Vector space

Fig. 1: Geometrical interpretation of the optimality conditions for intervals

2.5 Basic Geometrical Test

In this section, we outline the theoretical background of the Basic Geometrical Test described in [16] and introduce the necessary definitions and notations.

Definition 1 The inclusion of the conic hull **CH** for a coordinate i is defined as $\mathbf{CH}_i = \{d_i \mid d = (d_1, \ldots, d_n) \in \mathbf{CH}\}$, and the cone **CF** in the ith coordinate direction as $\mathbf{CF}_i = \{d_i \mid d \in \mathbf{CF}\}$, respectively. Furthermore, for coordinates i and j, the inclusion of the conic hull $\mathbf{CH}_{ij} = \{(d_i, d_j) \mid d \in \mathbf{CH}\}$ and the inclusion of the cone $\mathbf{CF}_{ij} = \{(d_i, d_j) \mid d \in \mathbf{CF}\}$ are defined similarly.

Note that $0 \notin \mathbf{CH}_k$ if $0 \notin \boldsymbol{\nabla}_k \mathbf{g}_j(\boldsymbol{x}), \forall j \in C$ and $0 \notin \boldsymbol{\nabla}_k \mathbf{p}_i(\boldsymbol{x}), \forall i \in B$. We also introduce the set-valued sign function in the following definition.

Definition 2 The set-valued sign function is a function, $\text{sign} : \mathbb{I} \to S \subseteq \{\text{"}-\text{"}, 0, \text{"}+\text{"}\}$, which categorizes the possible values in an interval $\boldsymbol{z}$, i.e.

$$\begin{aligned} 0 \in \text{sign}(\boldsymbol{z}), &\quad \textit{if } 0 \in \boldsymbol{z}, \\ \text{"}+\text{"} \in \text{sign}(\boldsymbol{z}), &\quad \textit{if } \overline{\boldsymbol{z}} > 0, \\ \text{"}-\text{"} \in \text{sign}(\boldsymbol{z}), &\quad \textit{if } \underline{\boldsymbol{z}} < 0. \end{aligned}$$

For an intervalbox $\boldsymbol{x}$, we define $\text{sign}(\boldsymbol{x}) = (\text{sign}(\boldsymbol{x}_1), \text{sign}(\boldsymbol{x}_2), \ldots, \text{sign}(\boldsymbol{x}_n))$.

Example 1 For the interval $([0, 13], [-2, 3], [0, 0])$, the set-valued sign function gives ("0+", "0+–","0"), as the first dimension is non-negative and the second is undetermined (it has positive and negative parts).

Proposition 1-3 and Corollary 1 are proven in [16]. Proposition 1-2 highlights that when the cone **CF** or conic hull **CH** contains all directions, they are full, and thus, by the optimality conditions, we cannot discard or reduce the intervalbox.

Proposition 1 ([16]) *If* $0 \in \nabla \mathbf{f}(\boldsymbol{x})$*, then, considering the inclusion of the gradients, the intervalbox* $\boldsymbol{x}$ *cannot be discarded by the optimality conditions.*

Proposition 2 ([16]) *If* $\exists\, i \in C : 0 \in \operatorname{int}(\boldsymbol{\nabla} \mathbf{g}_i(\boldsymbol{x}))$*, then the box* $\boldsymbol{x}$ *cannot be discarded by the optimality conditions, considering the inclusion of the gradients.*

Analyzing the cones coordinate-wise (see Definition 1), one can eliminate the box by easy-to-check conditions. Let us recall that $N = \{1, \ldots, n\}$.

Proposition 3 ([16]) *If* $\exists\, i \in N$ *such that* $\mathbf{CF}_i \cap \mathbf{CH}_i = \emptyset$*, then* $\nexists\, x^* \in \boldsymbol{x}$*, for which the optimality conditions are satisfied.*

The consequence of Proposition 3 is that the coordinate-wise cones – which can only take values $(-\infty, \infty), (-\infty, 0), (0, \infty), (-\infty, 0]$ or $[0, \infty)$ – can be examined using the set-valued sign function (see Definition 2).

Corollary 1 ([16]) If $\exists\, i \in N$ such that $\operatorname{sign}(\mathbf{CF}_i) \cap \operatorname{sign}(\mathbf{CH}_i) = \emptyset$ then the box cannot contain the global (or local) optimizer.

3 Theoretical Results

In this section, we present the notations and the theoretical results designed to refine the geometrical analysis of the interval optimality conditions. We introduce the matrix $\mathbf{G}$, which contains the gradient enclosures of the active constraints, i.e.

$$\mathbf{G} = \begin{bmatrix} \boldsymbol{\nabla}_1 \mathbf{g}_i(\boldsymbol{x}) & \ldots & \boldsymbol{\nabla}_1 \mathbf{p}_j(\boldsymbol{x}) \\ \vdots & \ddots & \vdots \\ \boldsymbol{\nabla}_n \mathbf{g}_i(\boldsymbol{x}) & \ldots & \boldsymbol{\nabla}_n \mathbf{p}_j(\boldsymbol{x}) \end{bmatrix}, \tag{8}$$

where $i \in C$ (the set of active constraints) and $j \in B$ (the set of active bound constraints). For instance, $\mathbf{G}_{kl}$ is the inclusion of the gradient of the active constraint l in dimension k. We define the dimensional gradient hull $\mathbf{GH}_k$, which is the interval hull of the gradients' inclusions for all active constraints in dimension k. Formally, this is given by $\mathbf{GH}_k = \bigsqcup_{i \in C, j \in B} (\boldsymbol{\nabla}_k \mathbf{g}_i(\boldsymbol{x}), \boldsymbol{\nabla}_k \mathbf{p}_j(\boldsymbol{x}))$. In other words, $\mathbf{GH}_k$ is equal

to the interval hull of row k in the matrix $\mathbf{G}$. Furthermore, $\mathbf{GH}_k \subseteq \mathbf{CH}_k$ because, denoting e_j as the jth unit vector,

$$\begin{aligned}\mathbf{GH}_k &= \bigsqcup_{i\in C, j\in B} \left(\boldsymbol{\nabla}_k \mathbf{g}_i(\boldsymbol{x}), \boldsymbol{\nabla}_k \mathbf{p}_j(\boldsymbol{x})\right) = \bigsqcup_{\substack{\mu=e_j \\ j\in C\cup B}} \left(\sum_{i\in C} \mu_i \boldsymbol{\nabla}_k \mathbf{g}_i(\boldsymbol{x}) + \sum_{i\in B} \mu_i \boldsymbol{\nabla}_k \mathbf{p}_i(\boldsymbol{x})\right) \\ &= \left\{ d_k \;\middle|\; d \in \sum_{i\in C} \mu_i \boldsymbol{\nabla} \mathbf{g}_i(\boldsymbol{x}) + \sum_{i\in B} \mu_i \boldsymbol{\nabla} \mathbf{p}_i(\boldsymbol{x}),\ \sum_{j\in C\cup B} \mu_j = 1 \right\} \\ &\subseteq \mathbf{CH}_k = \left\{ d_k \;\middle|\; d \in \sum_{i\in C} \mu_i \boldsymbol{\nabla} \mathbf{g}_i(\boldsymbol{x}) + \sum_{i\in B} \mu_i \boldsymbol{\nabla} \mathbf{p}_i(\boldsymbol{x}), \mu \geq 0, \mu \neq 0 \right\}.\end{aligned}$$

The coordinate-wise gradient hull vector is $\mathbf{GH} = (\mathbf{GH}_1, \ldots, \mathbf{GH}_n)$. We define the following coordinate sets for $\mathbf{GH}$:

Undetermined coordinate set U contains coordinate i if $\mathbf{GH}_i$ has both negative and positive elements, i.e. $U = \{i \in N \mid \text{sign}(\mathbf{GH}_i) = \text{"0+–"}\}$;
Determined coordinate set D contains coordinate i if $\mathbf{GH}_i$ does not have both negative and positive elements but needs either of them. That is, $D = \{i \in N \mid \text{sign}(\mathbf{GH}_i)$ is "+" or "0+" or "–" or "0–"$\}$;
Independent coordinate set I contains coordinate i if $\mathbf{GH}_i = 0$, so $I = \{i \in N \mid \text{sign}(\mathbf{GH}_i) = \text{"0"}\}$.

For Example 1, the above defined coordinate sets are the following: $U = \{2\}$, $D = \{1\}$, and $I = \{3\}$. To make things easier, we also introduce the notation $\mathbf{F}$ for $-\boldsymbol{\nabla}\mathbf{f}(\boldsymbol{x})$, so $\mathbf{F}_i = -\boldsymbol{\nabla}_i \mathbf{f}(\boldsymbol{x})$. Additionally, $\mathbf{F}_i \subseteq \mathbf{CF}_i$ as restricting $\mu_0 = 1$ in $\mathbf{CF}_i$ gives back $\mathbf{F}_i$.

Proposition 4 highlights that if there exists an independent coordinate k and a determined coordinate l, then the inclusion of the gradient of the objective function must contain zero in that coordinate for the optimality conditions to hold.

Proposition 4 *If* $\exists\, k, l \in N$ *such that* $\mathbf{GH}_k = 0$, $0 \notin \mathbf{F}_k$ *and* $0 \notin \mathbf{GH}_l$, *then* $\nexists\, x^* \in \boldsymbol{x}$, *for which the optimality conditions are satisfied.*

Proof Indirectly, suppose that the optimality conditions hold with the given assumptions. Thus, the equation $(k+1)$ of the NFJ conditions should hold, i.e.

$$\boldsymbol{\mu}_0 \boldsymbol{\nabla}_k \mathbf{f}(\boldsymbol{x}) + \sum_{i\in B} \boldsymbol{\mu}_i \boldsymbol{\nabla}_k \mathbf{p}_i(\boldsymbol{x}) + \sum_{j\in C} \boldsymbol{\mu}_j \boldsymbol{\nabla}_k \mathbf{g}_j(\boldsymbol{x}) = 0.$$

It simplifies to $\boldsymbol{\mu}_0 \boldsymbol{\nabla}_k \mathbf{f}(\boldsymbol{x}) = 0$, because from $\mathbf{GH}_k = 0$ the constraints gradient part,

$$\sum_{i\in B} \boldsymbol{\mu}_i \boldsymbol{\nabla}_k \mathbf{p}_i(\boldsymbol{x}) + \sum_{j\in C} \boldsymbol{\mu}_j \boldsymbol{\nabla}_k \mathbf{g}_j(\boldsymbol{x}) = 0.$$

Furthermore, $0 \notin \mathbf{F}_k$, thus $0 \notin \boldsymbol{\nabla}\mathbf{f}(\boldsymbol{x})$. The simplified equation $\boldsymbol{\mu}_0 \boldsymbol{\nabla}_k \mathbf{f}(\boldsymbol{x}) = 0$ holds only if $\boldsymbol{\mu}_0 = 0$. However, since $\boldsymbol{\mu} \geq 0$ and $\boldsymbol{\mu} \neq 0$, $\boldsymbol{\mu}_0 = 0$ is possible only if there is a $\boldsymbol{\mu}_j > 0$ for some $j \in B \cup C$. In this case, the stationarity condition reduces to $\sum_{i\in B} \boldsymbol{\mu}_i \boldsymbol{\nabla}_o \mathbf{p}_i(\boldsymbol{x}) +$

$\sum_{j \in C} \boldsymbol{\mu}_j \nabla_o \mathbf{g}_j(\boldsymbol{x}) = 0$ for all $o \in N$. This can only occur if for all coordinates $o \in N$, $0 \in \nabla_o \mathbf{g}_j(\boldsymbol{x})$ for any $j \in C$ or $0 \in \nabla_o \mathbf{p}_i(\boldsymbol{x})$ for any $i \in B$. This implies $0 \in \mathbf{GH}_o$ for all coordinates $o \in N$. However, it contradicts $0 \notin \mathbf{GH}_l$ in our assumptions. Therefore, $\nexists\, x^* \in \boldsymbol{x}$ for which the optimality conditions are satisfied. □

Remark 1 In Proposition 4, if there is no dimension $l \in N$ such that $0 \notin \mathbf{GH}_l$, but the remaining assumptions hold, then $\boldsymbol{\mu}_0$ must be zero. Therefore, we can verify whether a non-trivial linear combination of the inclusions of the gradients of the constraints, with non-negative multipliers $\boldsymbol{\mu}$, contains zero. If there is no such linear combination, the optimality conditions cannot hold, thus the box can be discarded. Note that this test checks whether the Linear Independence Constraint Qualification (LICQ) [21] holds.

The following proposition highlights that if all orthants of the vector space contain some part of the inclusions of the gradients of the active constraints, then the optimality conditions always hold. An orthant can be defined as $\mathcal{O}_s = \{x \in \mathbb{R}^n \mid s_i \cdot x_i \geq 0, \forall i \in N\}$ where $s = (s_1, s_2, \ldots, s_n) \in \{+1, -1\}^n$ indicates the signs of the orthant.

Proposition 5 *If for all orthants $\mathcal{O}_s, s \in \{+1, -1\}^n$ there exists $i \in C$ such that $\text{int}(\nabla \mathbf{g}_i(\boldsymbol{x}) \cap \mathcal{O}_s) \neq \emptyset$, then the intervalbox $\boldsymbol{x}$ cannot be discarded by the optimality conditions.*

Proof To prove our statement, it suffices to show that $\mathbf{CH} \cap \mathbf{CF} = \mathbf{CF}$, because then all points in the box $\boldsymbol{x}$ are possible local minimizers, taking into account only the inclusions of the gradients.

Given that for all orthants $\mathcal{O}_s, s \in \{+1, -1\}^n$ there exists $i \in C$ such that $\text{int}(\nabla \mathbf{g}_i(\boldsymbol{x}) \cap \mathcal{O}_s) \neq \emptyset$,, then all orthants contain some gradients. Therefore, the conic hull is full, i.e $\mathbf{CH} = \mathbb{R}^n$. Thus, whatever $\mathbf{CF}$ looks like, $\mathbf{CF} \subseteq \mathbf{CH}$, hence $\mathbf{CH} \cap \mathbf{CF} = \mathbf{CF}$ is what we aimed to show. □

Remark 2 Opposed to Proposition 5, if at least one coordinate is non-negative, non-positive or zero ($D \neq \emptyset$ or $I \neq \emptyset$), then the conic hull $\mathbf{CH}$ might not be full. Thus, if such a coordinate is found, then Proposition 5 cannot hold.

Proposition 6-7 highlight that the inclusion of the gradient of the objective function $\mathbf{F}_i$ (or the interval hull $\mathbf{GH}_i$) can be reduced if the interval hull $\mathbf{GH}_i$ (or the inclusion $\mathbf{F}_i$) is determined in coordinate i. This reduction will simplify our Advanced Geometrical Test described in Section 4 and is also useful in Propositions 8-9. For any set or interval $\mathbf{H}$, let $\mathbf{H}^+$ denote its non-negative part and $\mathbf{H}^-$ its non-positive part. Thus, $\mathbf{H}^+ \cup \mathbf{H}^- = \mathbf{H}$, where $\mathbf{H}^+ \cap \mathbf{H}^- = \mathbf{H} \cap \{0\}$.

Proposition 6 *If $\exists i \in N$ such that $\mathbf{F}_i \geq 0$ ($\mathbf{F}_i \leq 0$), then $\mathbf{GH}_i$ needs to be non-negative (non-positive) for the FJ optimality conditions to hold. So, we can reduce $\mathbf{GH}_i := \mathbf{GH}_i^+$ ($\mathbf{GH}_i := \mathbf{GH}_i^-$) without altering any solutions.*

Proof Suppose that $\mathbf{F}_i \geq 0$, thus $\mathbf{CF}_i \geq 0$. Therefore, $0 \leq \mathbf{CF}_i \cap \mathbf{CH}_i = (\mathbf{CF}_i \cap \mathbf{CH}_i)^+ = \mathbf{CF}_i \cap \mathbf{CH}_i^+$. As $\mathbf{GH}_i \subseteq \mathbf{CH}_i$, we can set $\mathbf{GH}_i := \mathbf{GH}_i^+$ without altering any solutions, since $\mathbf{GH}_i$ must be non-negative for optimality to hold.

If $\mathbf{F}_i \leq 0$, then $\mathbf{GH}_i$ must be non-positive, and we can reduce $\mathbf{GH}_i$ to $\mathbf{GH}_i^-$. The proof for this statement is very similar to the previous one. □

Proposition 7 *If there exists an $i \in N$ such that $\mathbf{GH}_i \geq 0$ ($\mathbf{GH}_i \leq 0$), then $\mathbf{F}_i$ must be non-negative (non-positive) to satisfy the FJ optimality conditions. Thus, we can safely reduce $\mathbf{F}_i$ to $\mathbf{F}_i^+$ ($\mathbf{F}_i^-$) without removing any possible local optimal solution using the reduced $\mathbf{F}_i$.*

Proof Similarly to the previous proof, suppose that $\mathbf{GH}_i \geq 0$, so $\mathbf{CH}_i \geq 0$. If $\mathbf{CH}_i \geq 0$, then $\mathbf{CF}_i \cap \mathbf{CH}_i \geq 0$, so $\mathbf{CF}_i \cap \mathbf{CH}_i = \mathbf{CF}_i \cap \mathbf{CH}_i^+$. As $\mathbf{F}_i \subseteq \mathbf{CF}_i$, we can set $\mathbf{F}_i := \mathbf{F}_i^+$ without altering any solution when using the reduced $\mathbf{F}_i$.

Similarly, if $\mathbf{GH}_i \leq 0$, then $\mathbf{F}_i := \mathbf{F}_i^-$ is also valid, as $\mathbf{F}_i$ needs to be non-positive for the optimality conditions to hold. □

Next, we show how non-optimality can be checked by aggregating the gradient enclosures of all active constraints into a single hull Lagrange multiplier $\boldsymbol{\lambda} = \bigcap_{i \in N} \frac{\mathbf{F}_i}{\mathbf{GH}_i}$. In the special case of a single active constraint and $0 \notin \mathbf{GH}$, the proposition also provides a direct enclosure of the Lagrange multiplier.

Proposition 8 *If, using the extended interval division,*

$$\bigcap_{i \in N} \frac{\mathbf{F}_i}{\mathbf{GH}_i} = \emptyset,$$

then $\nexists\, x^* \in \boldsymbol{x}$, for which the optimality conditions are satisfied. If $|B \cup C| = 1$ and $0 \notin \mathbf{GH}$, then

$$\boldsymbol{\mu}_0 = 1, \boldsymbol{\mu}_1 = \bigcap_{i \in N} \frac{\mathbf{F}_i}{\mathbf{GH}_i}$$

are valid inclusions for the Lagrange multipliers.

Proof Indirectly, let us suppose that there exists an $x^* \in \boldsymbol{x}$ which satisfies the optimality conditions. That is, there exists an intervalbox $\boldsymbol{\mu} \in \mathbb{I}^n, \boldsymbol{\mu} \geq 0$ such that

$$-\boldsymbol{\mu}_0 \boldsymbol{\nabla} \mathbf{f}(\boldsymbol{x}) = \sum_{k \in B} \boldsymbol{\mu}_k \boldsymbol{\nabla} \mathbf{p}_k(\boldsymbol{x}) + \sum_{k \in C} \boldsymbol{\mu}_k \boldsymbol{\nabla} \mathbf{g}_k(\boldsymbol{x}).$$

In other words, $\mathbf{CF} = \mathbf{CH}$, which means that $\mathbf{CF}_i = \mathbf{CH}_i$ for each coordinate i. These equations are equivalent to

$$\boldsymbol{\nu}_0 \mathbf{F}_i = \boldsymbol{\nu}_1 \mathbf{GH}_i, \quad i \in N, \tag{9}$$

where $\boldsymbol{\nu} \in \mathbb{I}^2, \boldsymbol{\nu} \geq 0$ as any non-negative combination of the gradients can be obtained from the interval hull of them, i.e.

$$\mathbf{CH}_i = \left\{ d_i \;\middle|\; d \in \sum_{k \in B} \mu_k \boldsymbol{\nabla} \mathbf{p}_k(\boldsymbol{x}) + \sum_{k \in C} \mu_k \boldsymbol{\nabla} \mathbf{g}_k(\boldsymbol{x}), \mu \geq 0, \mu \neq 0 \right\}$$

$$= \left\{ d_i \;\middle|\; d \in \nu_1 \left(\sum_{k \in B} \mu_k \nabla \mathbf{p}_k(\boldsymbol{x}) + \sum_{k \in C} \mu_k \nabla \mathbf{g}_k(\boldsymbol{x}) \right), \mu \geq 0, \sum_{k \in B \cup C} \mu_k = 1, \nu_1 \geq 0 \right\}$$
$$= \left\{ d_i \;\middle|\; d \in \nu_1 \mathbf{GH}, \nu_1 \geq 0 \right\} = \boldsymbol{\nu}_1 \mathbf{GH}_i,$$

and $\mathbf{CF}_i = \boldsymbol{\nu}_0 \mathbf{F}_i$ for any $i \in N$.

The solution of the interval-valued system of equations (9) exists only if there is a $\boldsymbol{\lambda} := \frac{\boldsymbol{\nu}_1}{\boldsymbol{\nu}_0}$ fulfilling all equations

$$\boldsymbol{\lambda} = \frac{\mathbf{F}_i}{\mathbf{GH}_i} \qquad i \in N.$$

There is no problem if $0 \in \boldsymbol{\nu}_0$ because of the extended interval division. Therefore, $\boldsymbol{\lambda}$ can be calculated by

$$\boldsymbol{\lambda} = \bigcap_{i \in N} \frac{\mathbf{F}_i}{\mathbf{GH}_i}.$$

Notice that if we were to combine all constraints into one, its gradient enclosure would be $\mathbf{GH}$. Therefore, we can interpret $\boldsymbol{\lambda}$ as a combined or hull Lagrange multiplier.

Since we supposed that there exists an $x^* \in \boldsymbol{x}$ which satisfies the optimality conditions, $\boldsymbol{\lambda} \neq \emptyset$. However, it contradicts $\boldsymbol{\lambda} = \bigcap_{i \in N} \frac{\mathbf{F}_i}{\mathbf{GH}_i} = \emptyset$ in our assumptions. Therefore, $\nexists\, x^* \in \boldsymbol{x}$ for which the optimality conditions are satisfied.

To prove the second statement of the proposition, we investigate the case where $|C| = 1$. The proof when $|B| = 1$ can be shown in a similar way.

If $|C| = 1$, then $B = \emptyset$ because $|B \cup C| = 1$. Let $\mathbf{g}_k(\boldsymbol{x})$ be the only one active constraint. In this case, there are two Lagrange multipliers: $\boldsymbol{\mu}_0$ for the objective function and $\boldsymbol{\mu}_1$ for $\mathbf{g}_k(\boldsymbol{x})$. The optimality conditions are reduced to equations $-\boldsymbol{\mu}_0 \nabla \mathbf{f}(\boldsymbol{x}) = \boldsymbol{\mu}_1 \nabla \mathbf{g}_k(\boldsymbol{x})$.

In addition, the constraint qualifications always hold because $0 \notin \mathbf{GH}$ and $|C| = 1$. Thus, we can set $\boldsymbol{\mu}_0$ to 1 and the solution of the interval-valued system of equations is

$$\boldsymbol{\mu}_1 = \bigcap_{i \in N} \frac{-\nabla_i \mathbf{f}(\boldsymbol{x})}{\nabla_i \mathbf{g}_k(\boldsymbol{x})}.$$

This solution is exactly the same as $\bigcap_{i \in N} \frac{\mathbf{F}_i}{\mathbf{GH}_i}$ because $\mathbf{F}_i = -\nabla_i \mathbf{f}(\boldsymbol{x})$ and $\mathbf{GH}_i = \nabla_i \mathbf{g}_k(\boldsymbol{x})$ by the definition of $\mathbf{F}_i$ and $\mathbf{GH}_i$. When $|B| = 1$, the proof is the same using the corresponding element of $\nabla \mathbf{p}(\boldsymbol{x})$. □

A consequence of Proposition 8 is the following. If $\mathbf{F}_i$ contains zero in an undetermined coordinate i, then the enclosure of the hull Lagrange multiplier $\boldsymbol{\lambda}$ is $(-\infty, \infty)$ in this coordinate.

Corollary 2 If $\exists i \in U$ such that $0 \in \mathbf{F}_i$, then $\frac{\mathbf{F}_i}{\mathbf{GH}_i} = (-\infty, \infty)$. Therefore, coordinate i can be left out when testing Proposition 8 for the box.

In Proposition 9, we investigate the conic hull $\mathbf{CH}$ and the cone $\mathbf{CF}$ by the inclusion of the possible line slopes in coordinates i and j for $\mathbf{CH}_{ij}$ and $\mathbf{CF}_{ij}$. For the cones, if the intersection of the slope inclusions is empty, then the optimality conditions cannot hold. We denote and compute the slope inclusion for $\mathbf{CF}_{ij}$ as $\mathbf{m}_{ij}^f = \frac{\mathbf{F}_j}{|\mathbf{F}_i|}$ and for $\mathbf{CH}_{ij}$ as $\mathbf{m}_{ij}^g = \bigsqcup_{k \in B \cup C} \frac{\mathbf{G}_{jk}}{|\mathbf{G}_{ik}|}$, where $|\cdot|$ denotes the absolute value. Taking the

absolute value in the nominator transforms the cones to the non-negative half-space in coordinate i to ease the calculations without altering the solutions.

Proposition 9 *Let $i \in D, j \in D \cup U$ such that $i \neq j$. If $\mathbf{m}_{ij}^g \cap \mathbf{m}_{ij}^f = \emptyset$, then $\nexists\, x^* \in \boldsymbol{x}$, for which the optimality conditions are satisfied.*

Proof Indirectly, let us suppose that there exists an $x^* \in \boldsymbol{x}$ which satisfies the optimality conditions. That is, there exists an intervalbox $\boldsymbol{\mu} \in \mathbb{I}^n, \boldsymbol{\mu} \geq 0$ in coordinates j and i such that

$$-\boldsymbol{\mu}_0 \boldsymbol{\nabla}_j \mathbf{f}(\boldsymbol{x}) = \sum_{k \in B} \boldsymbol{\mu}_k \boldsymbol{\nabla}_j \mathbf{p}_k(\boldsymbol{x}) + \sum_{k \in C} \boldsymbol{\mu}_k \boldsymbol{\nabla}_j \mathbf{g}_k(\boldsymbol{x}) \tag{10}$$

$$-\boldsymbol{\mu}_0 \boldsymbol{\nabla}_i \mathbf{f}(\boldsymbol{x}) = \sum_{k \in B} \boldsymbol{\mu}_k \boldsymbol{\nabla}_i \mathbf{p}_k(\boldsymbol{x}) + \sum_{k \in C} \boldsymbol{\mu}_k \boldsymbol{\nabla}_i \mathbf{g}_k(\boldsymbol{x}). \tag{11}$$

Equations (10) and (11) mean $\mathbf{CF}_j = \mathbf{CH}_j$ and $\mathbf{CF}_i = \mathbf{CH}_i$, respectively. These equations are equivalent to

$$\boldsymbol{\nu}_0 \mathbf{F}_j = \boldsymbol{\nu}_1 \mathbf{GH}_j \tag{12}$$

$$\boldsymbol{\nu}_0 \mathbf{F}_i = \boldsymbol{\nu}_1 \mathbf{GH}_i. \tag{13}$$

with $\boldsymbol{\nu} \in \mathbb{I}^2, \boldsymbol{\nu} \geq 0$, as shown in the proof of Proposition 8.

Notice that since $i \in D$, $\mathbf{GH}_i$ is non-positive or non-negative. Suppose that the conic hull and the cone lie in the same half-space in coordinates i. This occurs if Corollary 1 is not fulfilled. In this case, we can reflect the cones to the positive half-space in coordinate i without losing any solution. In formula,

$$\boldsymbol{\nu}_0 \mathbf{F}_j = \boldsymbol{\nu}_1 \mathbf{GH}_j \tag{14}$$

$$\boldsymbol{\nu}_0 |\mathbf{F}_i| = \boldsymbol{\nu}_1 |\mathbf{GH}_i|. \tag{15}$$

Furthermore, we express $\boldsymbol{\lambda} := \frac{\boldsymbol{\nu}_1}{\boldsymbol{\nu}_0}$ from equations (14)-(15). Thus,

$$\boldsymbol{\lambda} = \frac{\mathbf{F}_j}{\mathbf{GH}_j} \tag{16}$$

$$\boldsymbol{\lambda} = \frac{|\mathbf{F}_i|}{|\mathbf{GH}_i|}. \tag{17}$$

There is no problem if $0 \in \boldsymbol{\nu}_0$ due to the extended interval division. The solution of this interval-valued system of equations only exists if $\boldsymbol{\lambda}$ satisfies both (16) and (17). So, we can reduce it to a single equation

$$\frac{\mathbf{F}_j}{\mathbf{GH}_j} = \frac{|\mathbf{F}_i|}{|\mathbf{GH}_i|}. \tag{18}$$

Rearranging equation (18), we get

$$\frac{\mathbf{F}_j}{|\mathbf{F}_i|} = \frac{\mathbf{GH}_j}{|\mathbf{GH}_i|},$$

where $\frac{\mathbf{F}_j}{|\mathbf{F}_i|}$ is the inclusion of the line slopes of the cone, $\mathbf{m}_{ij}^f$, and $\frac{\mathbf{GH}_j}{|\mathbf{GH}_i|}$ is the inclusion of the line slopes of the conic hull, $\mathbf{m}_{ij}^g$.

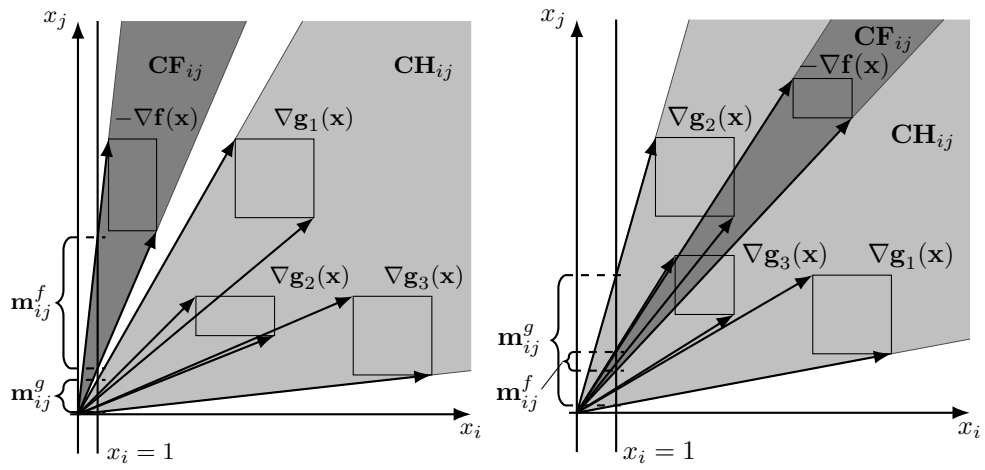
The fraction $\frac{\mathbf{GH}_j}{|\mathbf{GH}_i|}$ can be replaced by the interval hull $\bigsqcup_{k \in B \cup C} \frac{\mathbf{G}_{jk}}{|\mathbf{G}_{ik}|}$, which gives sharper bounds. The solution of the equation is $\mathbf{m}_{ij}^f \cap \mathbf{m}_{ij}^g$,which cannot be empty because there exists an $x^* \in \boldsymbol{x}$ that satisfies the optimality conditions. However, it contradicts $\mathbf{m}_{ij}^f \cap \mathbf{m}_{ij}^g = \emptyset$ in our assumptions. Therefore, $\nexists\, x^* \in \boldsymbol{x}$ for which the optimality conditions are satisfied. □

Remark 3 If $i \in D$ or $j \in D \cup U$ satisfies the assumption of Corollary 1, so the conic hull and the cone in coordinates i or j lie in opposite half-spaces. Therefore, Proposition 9 might not demonstrate the inexistence of an optimizer. However, the box can be removed by Corollary 1. Therefore, in the algorithm, Corollary 1 is checked first, followed by Proposition 9.

A useful consequence of Proposition 9 is the following. If $\mathbf{GH}_i$ and $\mathbf{F}_i$ contain zero for a given coordinate i, then the intersection of the conic hull and the cone is always non-empty. Therefore, it is pointless to check this coordinate.

Corollary 3 If $\exists i \in N$ such that $0 \in \mathbf{GH}_i$ and $0 \in \mathbf{F}_i$, then $\mathbf{m}_{ij}^g \cap \mathbf{m}_{ij}^f \neq \emptyset$ for any $j \in N$. Therefore, coordinate i is not used to discard the box.

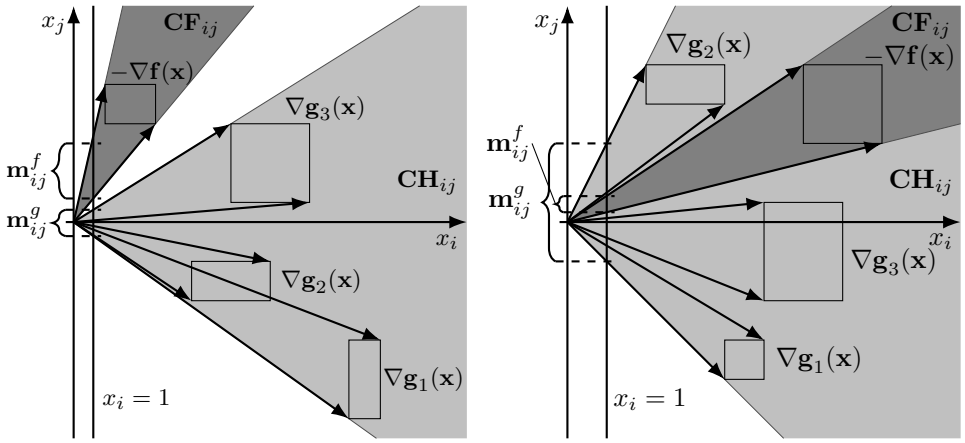
Figure 2 shows the optimality conditions for two determined coordinates. In Figure 2a, we can see the inclusion of the gradients of three active constraints in the coordinates i and j $(\boldsymbol{\nabla}\mathbf{g}_1(\boldsymbol{x}), \boldsymbol{\nabla}\mathbf{g}_2(\boldsymbol{x}), \boldsymbol{\nabla}\mathbf{g}_3(\boldsymbol{x}))$. We calculate the slope inclusions of the active constraints $\mathbf{m}_{ij}^g = \bigsqcup_{k\in\{1,2,3\}} \frac{\mathbf{G}_{jk}}{|\mathbf{G}_{ik}|} = \bigsqcup \left(\frac{[10,12]}{[8,12]}, \frac{[4,6]}{[6,10]}, \frac{[2,6]}{[14,18]} \right) = \left[\frac{1}{9}, \frac{3}{2}\right]$ and the slope inclusion of the objective $\mathbf{m}_{ij}^f = \frac{\mathbf{F}_j}{|\mathbf{F}_i|} = \frac{[8,12]}{[3/2,4]} = [2, 8]$. We can see that $\mathbf{m}_{ij}^g \cap \mathbf{m}_{ij}^f = \emptyset$. Thus, by Proposition 9, $\nexists\, x^* \in \boldsymbol{x}$, for which the optimality conditions are satisfied. Figure 2b shows a similar example where $\mathbf{m}_{ij}^f \subset \mathbf{m}_{ij}^g$. Therefore, in this case, we cannot discard the box due to the properties of these two coordinates.



(a) The optimality conditions cannot hold (b) The optimality conditions might hold

Fig. 2: Optimality conditions for two determined coordinates

Figure 3 shows the optimality conditions for one determined and one undetermined coordinate. We can always set the coordinates such that the first coordinate is the



(a) The optimality conditions cannot hold (b) The optimality conditions might hold

Fig. 3: Optimality conditions for one determined and one undetermined coordinate

determined one and the second coordinate is the undetermined one. This is useful because the non-continuity of the slopes for the vertical line can be avoided. Figure 3a shows the case where we can discard the box by Proposition 9. Figure 3b shows the case where $\mathbf{m}_{ij}^f \subset \mathbf{m}_{ij}^g$, so we cannot discard the box using these two coordinates.

4 Advanced Geometrical Test

Based on the theoretical framework and notation outlined in Sections 2.5 and 3, we propose the Advanced Geometrical Test. This preprocessing method is rigorous but incomplete: where a decision is reached, it is mathematically certified, however, it does not guarantee a decision in every case. While the derived propositions and corollaries are independently valid, their hierarchical application is crucial for algorithmic efficiency. The Advanced Geometrical Test has two distinct objectives: first, to discard boxes for which the FJ conditions provably cannot be satisfied; second, to skip the FJ evaluation entirely in cases where it is certified to yield no reduction. The structure is designed such that the computationally cheaper tests are performed first. Additional computation is only performed if a decision cannot be made. Consequently, the full or reduced FJ optimality system is solved only as a last resort when the box cannot be eliminated or reduced through these geometrical investigations. Note that even when the preliminary test does not reach a decision, executing the FJ test subsequently may still yield a reduction. The Advanced Geometric Test serves only to avoid unnecessary FJ evaluations, not to replace them.

In more detail, the Advanced Geometrical Test, shown in Algorithm 1, proceeds as follows. In line 1, we check the gradient $\boldsymbol{\nabla}\mathbf{f}(\boldsymbol{x})$. If this inclusion contains zero, then by Proposition 1, the box cannot be reduced. Therefore, we stop and skip solving the FJ optimality conditions. Note that for this box, all further evaluations in the optimality conditions are saved. In lines 3-13, we examine each active constraint $i \in C$. We

calculate the gradient's inclusion of the constraint, its centred form, and update the inclusion of the active constraint in line 5. If the updated inclusion is greater than zero, then the box is infeasible, and we discard it (lines 6-7). If the inclusion is less than zero, then the constraint is inactive, and we update the set of active constraints (lines 8-9). Otherwise, the constraint is still active, and we check the gradient $\boldsymbol{\nabla}\mathbf{g}_i(\boldsymbol{x})$. If the interior contains zero, then the box cannot be reduced by Proposition 2. Therefore, we stop and skip solving the FJ conditions, saving the evaluation of all remaining constraint's gradients. Otherwise, we store the inclusion of the gradient in a new column of the matrix $\mathbf{G}$.

Algorithm 1 Advanced Geometrical Test

Input: `X` the current node.
Output: `case` (discarded, skipFJ, solveFJ, solveReducedFJ, feasible).

1: **if** $0 \in \boldsymbol{\nabla}\mathbf{f}(\boldsymbol{x})$ **then** ▸ *Proposition 1: full cone* **CF**
2: **return** `case` $\leftarrow$ skipFJ;
3: **for** $i \in C$ **do**
4: Calculate $\boldsymbol{\nabla}\mathbf{g}_i(\boldsymbol{x})$
5: $\mathbf{g}_i(\boldsymbol{x}) \leftarrow \mathbf{g}_i(\boldsymbol{x}) \cap (\mathbf{g}_i(c) + \boldsymbol{\nabla}\mathbf{g}_i(\boldsymbol{x})(\boldsymbol{x} - c))$; ▸ *Update the inclusion of* $\mathbf{g}_i(\boldsymbol{x})$ *with centred form*
6: **if** $\mathbf{g}_i(\boldsymbol{x}) > 0$ **then** ▸ *Infeasible case* $(\mathbf{g}_i(\boldsymbol{x}) > 0)$
7: **return** `case` $\leftarrow$ discarded;
8: **else if** $\mathbf{g}_i(\boldsymbol{x}) < 0$ **then** ▸ *Strictly feasible case* $(\mathbf{g}_i(\boldsymbol{x}) < 0)$
9: $C \leftarrow C \setminus \{i\}$; ▸ *Delete index of the inactive constraint*
10: **else** ▸ *Undetermined case* $(0 \in \mathbf{g}_i(\boldsymbol{x}))$
11: **if** $0 \in \text{int}(\boldsymbol{\nabla}\mathbf{g}_i(\boldsymbol{x}))$ **then** ▸ *Proposition 2: full conic hull* **CH**
12: **return** `case` $\leftarrow$ skipFJ;
13: $\mathbf{G} \leftarrow [\mathbf{G}, \boldsymbol{\nabla}\mathbf{g}_i(\boldsymbol{x})]$; ▸ *Store the ith inclusion of the gradients in* $\mathbf{G}$ *(in the next column)*
14: $[B, \texttt{Bgrad}] \leftarrow \text{CalcActBoundConst}(\texttt{X}, \boldsymbol{y})$; ▸ *Determine the active bound constraints*
15: **if** $C = \emptyset$ **then** ▸ *Strictly feasible box* $(\forall i \in \{1, \ldots, m\} : \mathbf{g}_i(\boldsymbol{x}) < 0)$
16: **return** [`case` $\leftarrow$ feasible, `X`] $\leftarrow \text{MonotonicityTest}(\texttt{X}, B)$;
17: $\mathbf{G} \leftarrow [\mathbf{G}, \texttt{Bgrad}]$; $\mathbf{F} \leftarrow -\boldsymbol{\nabla}\mathbf{f}(\boldsymbol{x})$;
18: $[\texttt{case}, \texttt{sgF}, \texttt{sgG}, \texttt{sgGH}, D, U, I] \leftarrow \text{SignTest}(\mathbf{F}, \mathbf{G})$; ▸ *see Algorithm 2*
19: **if** `case` $\in \{\text{discarded}, \text{solveReducedFJ}\}$ **then** ▸ *Corollary 1, Proposition 4 or Remark 1 holds*
20: **return** `case`;
21: **if** $\text{CheckAllOrthantUsed}(\texttt{sgG}, D, I)$ **then** ▸ *Proposition 5: full conic hull* **CH**
22: **return** `case` $\leftarrow$ skipFJ;
23: **if** $|B \cup C| = 1$ **then** ▸ *One active constraint*
24: **if** $\text{EncloseHullLagMult}(\mathbf{F}, \mathbf{G}, \texttt{sgF}, D, U) = \emptyset$ **then** ▸ *see Algorithm 3*
25: **return** `case` $\leftarrow$ discarded;
26: **return** `case` $\leftarrow$ skipFJ;
27: **else** ▸ *Two or more active constraints*
28: **if** $\text{EncloseHullLagMult}(\mathbf{F}, \bigsqcup(\mathbf{G}), \texttt{sgF}, D, U) = \emptyset$ **then** ▸ *see Algorithm 3*
29: **return** `case` $\leftarrow$ discarded;
30: **if** $\text{DiscardByTwoDim}(D, U, \texttt{sgF}, \texttt{sgGH}, \mathbf{G}, \mathbf{F})$ **then** ▸ *see Algorithm 4*
31: **return** `case` $\leftarrow$ discarded;
32: **return** `case` $\leftarrow$ solveFJ;

We need to construct the matrix $\mathbf{G}$ and calculate the sets D, U and I in order to apply the remaining theoretical statements. Thus, in line 14, the indices and the gradient enclosures of the active bound constraints are stored in B and `Bgrad`. In lines 15-16, if it turns out that all constraints are inactive, then the box is strictly feasible. We perform the Monotonicity Test and stop the procedure. In line 17, we add the gradient columns of the active bound constraints to $\mathbf{G}$. In lines 18–20, we call the SignTest (see Section 4.1) to check the signs of the gradients, calculate the sets D, U and I verify Corollary 1, Proposition 4 and Remark 1. If the SignTest returns with

the 'cont' case, Proposition 5 is applied in the CheckAllOrthantUsed function in lines 21-22. So, we check that every orthant contains an inclusion of the gradient of the active constraints, which means that the conic hull **CH** is full. If it is full, we skip solving the FJ optimality conditions. Next, we differentiate the one-constrained case in lines 23-26 from the case when more active constraints are present. In the one active constraint case, in lines 24-25, we calculate the inclusion of the Lagrange multipliers using Proposition 8. The method is described in detail in Section 4.2. If the calculated inclusion is empty, we discard the box. Otherwise, we skip solving the FJ optimality conditions. In the case of two or more active constraints, in lines 28-29, we apply Proposition 8, detailed in Section 4.2. In lines 30-31, we examine whether the box can be discarded by two coordinates using Proposition 9. The methods are described in detail in Section 4.3. If the box cannot be discarded, we solve the FJ conditions.

4.1 Sign Test

The method is described in Algorithm 2. In order to apply Corollary 1, for each coordinate, we calculate the sign of the gradient of the objective function, the active constraints, and their interval hull (lines 3-6).

Algorithm 2 SignTest

Input: $\mathbf{F}, \mathbf{G}$
Output: `case`(discarded, solveReducedFJ, cont), $\mathtt{sgF}, \mathtt{sgG}, \mathtt{sgGH}, D, U, I$

1: `case` $\leftarrow$ cont, `isPossibleNonOpt` $\leftarrow$ False; $D \leftarrow \emptyset; U \leftarrow \emptyset; I \leftarrow \emptyset$;
2: **for** $i \in N$ **do**
3: $\quad \mathtt{sgF}_i \leftarrow \text{getSign}(\mathbf{F}_i)$; ▸ $\text{sign}(\boldsymbol{\nabla}_i \mathbf{f}(\boldsymbol{x}))$
4: $\quad$ **for** $j \in B \cup C$ **do**
5: $\quad\quad \mathtt{sgG}_{ij} \leftarrow \text{getSign}(\mathbf{G}_{ij})$; ▸ $\text{sign}(\boldsymbol{\nabla}_i \mathbf{g}_j(\boldsymbol{x}))$
6: $\quad \mathtt{sgGH}_i \leftarrow \text{unionSign}(\mathtt{sgG}_{ij}, j \in B \cup C)$; ▸ $\text{sign}(\mathbf{GH}_i)$
7: $\quad$ **if** $\mathtt{sgF}_i \cap \mathtt{sgGH}_i = \emptyset$ **then** ▸ *Corollary 1:* $\text{sign}(\mathbf{CF}_i) \cap \text{sign}(\mathbf{CH}_i) = \emptyset$
8: $\quad\quad$ **return** `case` $\leftarrow$ discarded;
9: $\quad$ **switch** $\mathtt{sgGH}_i$ **do**
10: $\quad\quad$ **case** "0+−" : $U \leftarrow U \cup \{i\}$; ▸ *Undetermined coordinate*
11: $\quad\quad$ **case** "0" : ▸ *Independent coordinate*
12: $\quad\quad\quad$ **if** $\mathtt{sgF}_i \in \{\text{"+"}, \text{"−"}\}$ **then** ▸ *Remark 1:* $\mathbf{GH}_i = 0$ *and* $0 \notin \mathbf{F}_i$
13: $\quad\quad\quad\quad$ `isPossibleNonOpt` $\leftarrow$ True;
14: $\quad\quad\quad I \leftarrow I \cup \{i\}$;
15: $\quad\quad$ **default** : $D \leftarrow D \cup \{i\}$; ▸ *Determined coordinate*
16: **if** `isPossibleNonOpt` = True **then**
17: $\quad$ **if** $\{l \mid l \in D, 0 \notin \mathtt{sgGH}_l\} \neq \emptyset$ **then** ▸ *Proposition 4:* $\mathbf{GH}_i = 0$, $0 \notin \mathbf{F}_i$ *and* $\exists l \in N : 0 \notin \mathbf{GH}_l$
18: $\quad\quad$ `case` $\leftarrow$ discarded;
19: $\quad$ **else**
20: $\quad\quad$ `case` $\leftarrow$ solveReducedFJ;
21: **return** `case`, $\mathtt{sgF}, \mathtt{sgG}, \mathtt{sgGH}, D, U, I$;

If the intersection of the signs is empty, we stop the procedure (the box can be discarded). In lines 9-15, we determine the independent, determined, and undetermined sets. If an independent coordinate is found and the inclusion of the objective function does not contain zero, the flag isPossibleNonOpt is set to true. In such a case, we check Proposition 4 in lines 16-20. If the proposition holds, the box is discarded. If it does not hold, the Lagrange estimator method is solved without the column of the objective function (case = solveReducedFJ). Otherwise, we continue the Advanced Geometrical

Test. Since Corollary 1 applies to a single dimension, it can be used within the loop. In contrast, Proposition 4 requires the conditions to be met across two dimensions. That is why it is applied later. Furthermore, although the assumptions of Remark 1 are weaker than those of Proposition 4, we check that later because it is computationally more demanding than testing Proposition 4.

4.2 Enclose the Hull Lagrange Multipliers Method

This method is described in Algorithm 3. First, we delete the ineffective coordinates from U by Corollary 2. Otherwise, we reduce the inclusion of $\mathbf{GH}_i$ by Proposition 6. If the number of relevant coordinates is greater than zero, we enclose the hull Lagrange multipliers in line 8.

Algorithm 3 Enclose Hull Lagrange Multipliers Method

Input: $\mathbf{F}, \mathbf{GH}, \mathtt{sgF}, D, U$
Output: $\boldsymbol{\lambda}$
1: $\boldsymbol{\lambda} \leftarrow (-\infty, \infty)$;
2: **for** $i \in U$ **do**
3: **if** $\mathtt{sgF}_i \in \{$"0", "0+–"$\}$ **then** ▸ *Corollary 2: remove ineffective coordinates*
4: $U \leftarrow U \setminus \{i\}$;
5: **else**
6: $\mathbf{GH}_i \leftarrow \text{reduce}(\mathbf{GH}_i, \mathtt{sgF}_i)$; ▸ *Proposition 6: reduce the hull gradient*
7: **if** $|U \cup D| > 0$ **then**
8: $\boldsymbol{\lambda} \leftarrow \cap_{i \in U \cup D} \frac{\mathbf{F}_i}{\mathbf{GH}_i}$; ▸ *Enclose the Hull Lagrange Multipliers*
9: **return** $\boldsymbol{\lambda}$;

4.3 Discarding By Two Coordinates Method

The method checks whether Proposition 9 holds. It is described in Algorithm 4. In lines 1–2, we pair the possible coordinates for two determined coordinates in DD and for one determined and one undetermined coordinate in DU using Corollary 3.

Algorithm 4 Discard By Two Coordinates Method

Input: $D, U, \mathtt{sgF}, \mathtt{sgGH}, \mathbf{G}, \mathbf{F}$
Output: `discard`
1: $DD \leftarrow \left\{(i,j) \mid i,j \in D, i < j, 0 \notin \mathtt{sgGH}_i, 0 \notin \mathtt{sgF}_i, 0 \notin \mathtt{sgF}_j, 0 \notin \mathtt{sgGH}_j\right\}$ ▸ *Corollary 3*
2: $DU \leftarrow \left\{(i,j) \mid i \in D, j \in U, 0 \notin \mathtt{sgGH}_i, 0 \notin \mathtt{sgF}_i, 0 \notin \mathtt{sgF}_j\right\}$ ▸ *Corollary 3*
3: **for** $(i,j) \in DD, (i,j) \in DU$ **do** ▸ *First all pairs in DD then the pairs in DU*
4: $\mathbf{F}_i \leftarrow \text{reduce}(\mathbf{F}_i, \mathtt{sgGH}_i)$; ▸ *Proposition 7: reduce the gradient of the objective function*
5: **if** $\bigsqcup_{k \in B \cup C} \frac{\mathbf{G}_{jk}}{|\mathbf{G}_{ik}|} \bigcap \frac{\mathbf{F}_j}{|\mathbf{F}_i|} = \emptyset$ **then** ▸ *Proposition 9:* $\mathbf{m}^g_{ij} \cap \mathbf{m}^f_{ij} = \emptyset$
6: **return** `discard` $\leftarrow$ True;
7: **return** `discard` $\leftarrow$ False;

In lines 3-6, we iterate through the paired coordinates. We reduce $\mathbf{F}_i$ by Proposition 7. This reduction is very useful because it ensures that the line slope $\mathbf{m}^f_{ij} \subset (-\infty, \infty)$. We calculate the slope inclusions for both the gradients of the active constraints and the objective function. In line 5, if Proposition 9 holds, we stop the procedure, and

the box is discarded. Once all possible pairs of coordinates have been examined, we set the variable `discard` to false and stop the procedure (line 7).

4.4 Additional ideas for the Advanced Geometrical Test

We explored several strategies to enhance efficiency, but we found them to be computationally ineffective. Firstly, in scenarios with a single active constraint, neither the full NFJ solution nor a reduced formulation exploiting $\boldsymbol{\mu}_0 = 1$ outperformed simply skipping the test. Secondly, the DiscardByTwoDim method attempted to pair undetermined coordinates and compare gradient slopes. However, the complex slope calculations and frequently full conic hulls resulted in an overall increase in running time. Finally, we trained decision tree models to predict the utility of optimality checks based on geometric features. While these models achieved high classification accuracy with low complexity, the time spent on feature extraction was greater than the time saved by skipping unsuccessful checks. See Appendix C for detailed descriptions of these additional ideas.

5 Computational experiments

We implemented the IBB method from Section 2.3, combining the Normalized Interval Fritz-John Optimality Condition System (Section 2.2) and the Advanced Geometrical Test (Section 4). In this section, we experimentally evaluate the efficiency of the Advanced Geometrical Test (AdvTest).

5.1 Environment and parameters

We used MATLAB R2020a [22] with INTLAB 11[18], employing only its Interval Arithmetic, Automatic Differentiation, and Interval LU decomposition modules. Notably, INTLAB implements automatic differentiation efficiently: it skips computations for variables that do not affect the function and stores gradients and Hessians as sparse structures.

In the IBB method described in Section 2.3, we used the following parameters: a time limit of 3600 seconds, an accuracy of 10^{-4} for both the inclusion function and the box width, and a non-deformed ratio threshold of 0.5. Table 1 lists the abbreviations for the nine tested methods. All tests were executed on an AMD Ryzen 7 3800X 8-Core processor with 32 GB of RAM.

We used a generated benchmark containing a total of 374 test cases. The benchmark was generated from 34 well-known unconstrained test cases, listed in Table 2, taken from [2, 23] with the same search region mentioned in those papers. For each function, 2, 4, or 6 constraints were generated. The test cases are split into two categories: those where the optimizer points are the same as in the unconstrained case (IN problems) and those where the unconstrained optimizer points are cut from the search region (OUT problems). The IN problems originate from paper [16], in which the generation of the constraints is described in detail. This set contains a total of 272 test cases, categorized by the number of coordinates, the number of constraints, and the number of active constraints in the optimum, which can be 0, 2, or 4.

Table 1: Abbreviations of the methods.

Abbreviations	Methods
IBB	IBB without FJ optimality conditions
Lag	IBB with the Lagrange estimator method
LFJ	IBB with the Lagrange estimator + NFJ method
Geo+IBB	IBB with Geometrical Test without using FJ optimality conditions
Geo+Lag	IBB with Geometrical Test and the Lagrange estimator method
Geo+LFJ	IBB with Geometrical Test and the Lagrange estimator + NFJ method
Adv+IBB	IBB with Advanced Geometrical Test without FJ optimality conditions
Adv+Lag	IBB with Advanced Geometrical Test and the Lagrange estimator method
Adv+LFJ	IBB with Advanced Geometrical Test and the Lagrange estimator+NFJ method

Table 2: The 34 original bound constrained test cases from [2, 23].

Dimension	Test cases and abbreviations
2	Rosenbrock(RB2), Six-Hump-Camel-Back(SHCB), The 2002 SIAM Challenge #4(SIAM), Branin(BR), Levy-3(L3), Levy-5(L5), Levy-13(L13), Schwefel-2.1(SW21), Schwefel-2.5(SW25), Ratz-4(R4), Schwefel-2.18(SW218), Three-Hump-Camel-Back(THCB), Goldstein-Price(GP)
3	Hartmann-3(H3), Levy-8(L8), Levy-14(L14), Schwefel-3.1(SW31), Schwefel-3.2(SW32)
4	Levy-9(L9), Levy-15(L15), Shekel-5(S5), Shekel-7(S7), Shekel-10(S10)
5	5-dimensional-Rosenbrock(RB5), Griewank-5(G5), Levy-10(L10), Levy-16(L16), Schwefel-3.7.5(SW375)
6–10	Griewank-7(G7), Hartmann-6(H6), Schwefel-3.7.10(SW3710), Levy-11(L11), Levy-12(L12), Levy-18(L18)

The OUT set of problems contains 102 test cases, where for all problems in Table 2 we generated 2, 4, or 6 constraints in a similar way as for the IN set, but cutting out the optimizer points. Unlike with IN problems, we cannot predict the number of active constraints at the new minimizer points in advance. For the generation of these problems, only the constraint generator procedure was modified. We randomly select a point x^* from OPT, the set of optimizers of the unconstrained problem, and generate an active constraint at this point. We shift the constraint a bit in a direction related to the gradient at this point or randomly to cut x^* from the feasible region. Then, we remove the optimizer from OPT and generate the next constraint until the set OPT becomes empty. The created test cases with figures for the two-dimensional problems and the code written in Matlab can be found in the GitLab repository[1].

5.2 Comparison with other global and local solvers

We benchmarked our proposed method against eight nonlinear solvers using the generated test cases. This included three global solvers (Baron 25.8.5, LGO 2015-01-17 and LINDO 13.0.4099.255) and five local solvers (CONOPT 4.36, KNITRO 14.2.0, LOQO 7.03, MINOS 5.51 and SNOPT 7.5-1.2) in the AMPL environment. To match

[1]IBB with Optimality conditions - Intlab:
https://gitlab.com/gencsimiska27/the-ibb-with-geo-and-fj-test-intlab

the IBB method, we configured the solvers by setting the absolute and relative convergence tolerances to 10^{-4} and 10^{-10} respectively, and the runtime limit to 3 600 seconds. All other settings were left at their defaults.

The performances of the global and local solvers are detailed in Tables 3–4. The result is organized into three groups and reports the percentage of the test cases. The first group details the raw outcomes reported by the solver: 'Solved' (optimal solution found), 'Solved?' (likely optimal, possible error), 'Limit' (resource limit reached), 'Infeasible', and 'Failure' (error condition). The second group compares the solver's results, For IN test cases, where the global optimum is known, a solution is classified as 'Optimal' if the absolute error is below 10^{-4}. For OUT test cases, where the global optimum is unknown, we check whether the solver's result lies within the best computed inclusion interval of the objective function. Results that violate constraints are marked as 'Infeasible'. Otherwise, they are marked as 'Non-optimal'. Finally, the third group counts the number of times each solver failed to find the global optimum (returning a non-optimal value or infeasible).

Table 3 presents the performance of the global solvers. The Baron solver could only solve half of the test cases because it cannot handle trigonometric functions. While the LGO solver processed most of the cases, it frequently returned an ambiguous status ('Solved?'). LINDO successfully solved the majority of test cases, but 7.2% remained unsolved within the specified time limit ('Limit'). Verification of the reported results revealed significant discrepancies: some of the returned objective values were local optima, and certain points were infeasible. Examining the last column shows that the 'Solved' status does not always mean that the problem has been solved correctly (i.e. it is non-optimal or infeasible). In contrast, our proposed method (Adv+Lag) solved 94.1% of the test cases, while the remaining cases stopped due to the time limit.

Table 3: Performance of the global solvers.

Solver	Solved	Solved?	Limit	Infeasible	Failure	Non-optimal	Infeasible	Optimal	Solved& Incorrect
Baron	47.3%	0.0%	2.7%	0.0%	50.0%	0.0%	11.0%	36.4%	11.0%
LGO	23.0%	77.0%	0.0%	0.0%	0.0%	13.6%	21.7%	64.7%	13.1%
LINDO	92.8%	0.0%	7.2%	0.0%	0.0%	9.6%	17.4%	73.0%	19.8%
Own method	94.1%	0.0%	5.9%	0.0%	0.0%	0.0%	0.0%	100.0%	0.0%

Table 4 presents the performance of the local solvers. While most solvers (particularly KNITRO and LOQO) marked a large percentage of runs as successful ("Solved"), a substantial error rate was observed during validation. The final column of the table ('Solved & Incorrect') quantifies this unreliability. For example, SNOPT successful runs 60.7% were marked as incorrect. CONOPT did not mark any test cases as 'Solved', but it produced many uncertain results.

Based on the verified results, the floating-point global solvers identified the global optimum in 36–73% of cases, and the local solvers in 25–50%. This is primarily due to the accumulation of rounding errors and numerical noise. A critical deficiency that has been observed is that, despite successful convergence being reported, infeasible points

Table 4: Performance of the local solvers.

Solver	Solved	Solved?	Limit	Infeasible	Failure	Non-optimal	Infeasible	Optimal	Solved& Incorrect
CONOPT	0.0%	87.2%	0.0%	12.8%	0.0%	32.9%	27.3%	39.8%	0.0%
KNITRO	100.0%	0.0%	0.0%	0.0%	0.0%	44.4%	5.6%	50.0%	50.0%
LOQO	94.9%	0.0%	5.1%	0.0%	0.0%	50.8%	9.9%	39.3%	55.6%
MINOS	66.8%	0.5%	13.1%	19.5%	0.0%	50.3%	24.3%	25.4%	52.4%
SNOPT	87.4%	0.8%	1.9%	9.9%	0.0%	34.0%	29.7%	36.4%	60.7%

are often returned by these solvers. Furthermore, their search mechanisms are often limited and tend to converge on a single local minimum rather than finding all global optimal points. By contrast, our approach is designed to overcome these numerical instabilities. It successfully identifies all global minima within the search space while strictly maintaining feasibility.

5.3 Comparison of the variants of the IBB for the IN and OUT problems

Figure 4 presents a box plot of running times (in seconds, on a logarithmic scale) for all test cases, grouped into IN and OUT categories. The symbol ✖ indicates the arithmetic mean (AM), while the symbol ✚ represents the shifted geometric mean (SGM). The SGM is computed using the formula

$$SGM_m(x) = \sqrt[m]{\prod_{i=1,\ldots,m} (x_i + s)} - s, \tag{19}$$

where m is the number of problems and s is equal to 10.

In Figure 4, the results show that, in general, the IN problems are much easier to solve than the OUT problems. Also, the basic IBB method and Geo+IBB perform much better on the IN set, and the use of optimality conditions, Adv, Lag, or LFJ helps more in the OUT cases. For the OUT cases, the Lag method demonstrates better performance than the LFJ method unless AdvTest is also used. Nevertheless, in both categories, the methods with the AdvTest perform best in both shifted geometric and arithmetic means.

Figure 5 shows the percentage of solved problems for each method within the given time shown on the horizontal axis. As noted previously, all methods perform better on IN test cases than on OUT test cases. In the IN category, there is a clear distinction in performance between the methods with and without the AdvTest. Specifically, methods that include the AdvTest solve a significantly higher percentage of problems. In the OUT category, although the overall success rates are lower, we still observe a general increase in performance over time. Introducing the Lagrange estimator + NFJ (LFJ) method into these methods improves their performance by approximately 10%, as reflected in the next two lines above them. Replacing the LFJ method with the Lag method yields further efficiency gains. The top three curves correspond to methods

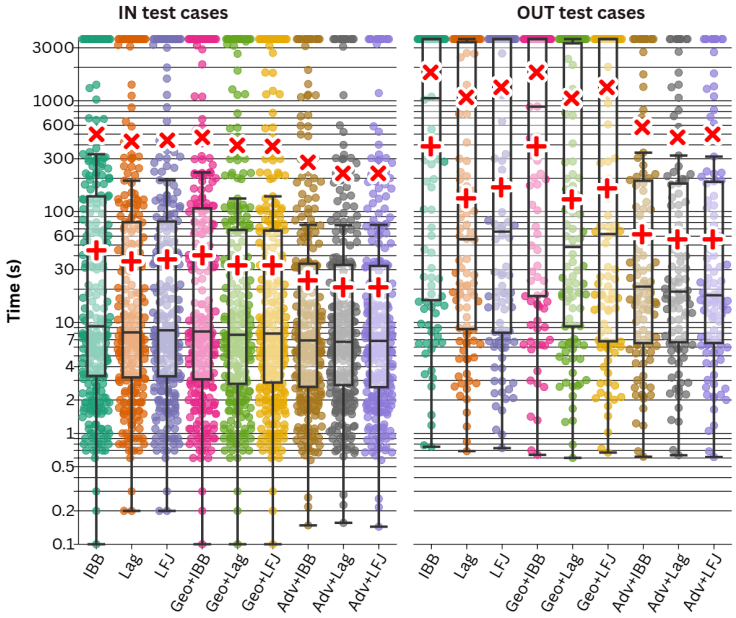


Fig. 4: The box plot of the running times in seconds on logarithmic scale for all problems grouped into IN and OUT categories.

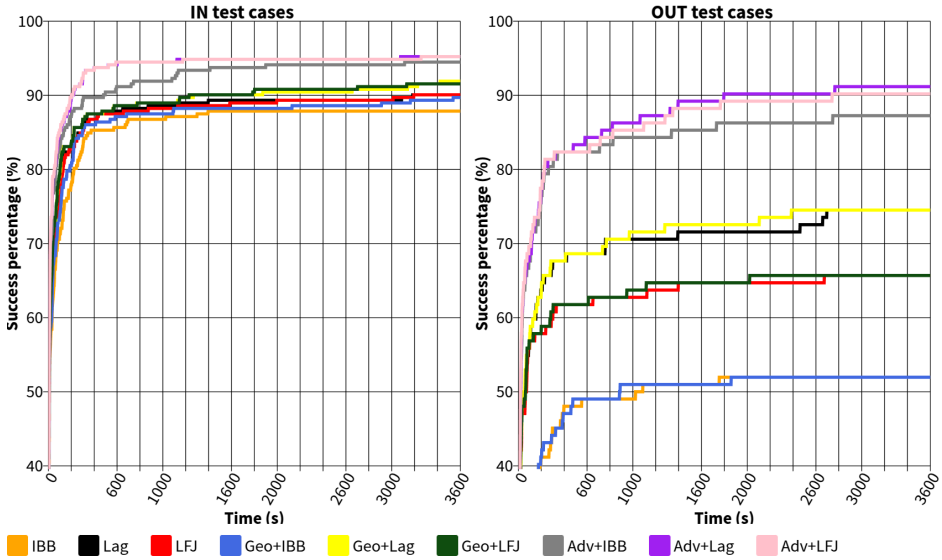


Fig. 5: The percentage of solved problems grouped to IN and OUT sets.

that include the AdvTest. However, the absolute performance is notably higher for IN tasks, reaffirming that they are generally easier to solve.

5.4 Comparison of the variants of the IBB method

Figure 6–8 illustrate box plots with whiskers, summarizing the performance of all methods across the full set of test cases. In each plot, the symbol ✖ denotes the arithmetic mean (AM) and the symbol ✚ the shifted geometric mean (SGM), which are calculated using the formula given in (19).

Figure 6 shows a box plot of running times (in seconds, on a logarithmic scale) for test cases completed within one hour. The best performing methods are those with the AdvTest in terms of shifted geometrical and arithmetic mean.

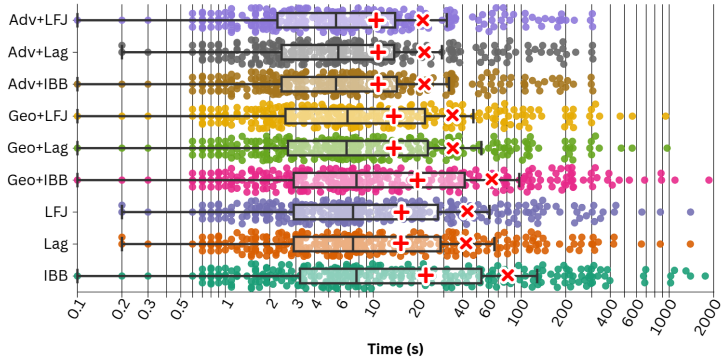


Fig. 6: The box plot of the running times in seconds on logarithmic scale for all problems finishing in one hour.

Figure 7 shows a box plot of the running times (in seconds) of the test cases for which at least one method cannot solve the problem within one hour. As previously observed, the methods with the AdvTest are the best performers. These methods are significantly more efficient than those without it, demonstrating the effectiveness of advanced geometric reasoning in tackling more difficult instances.

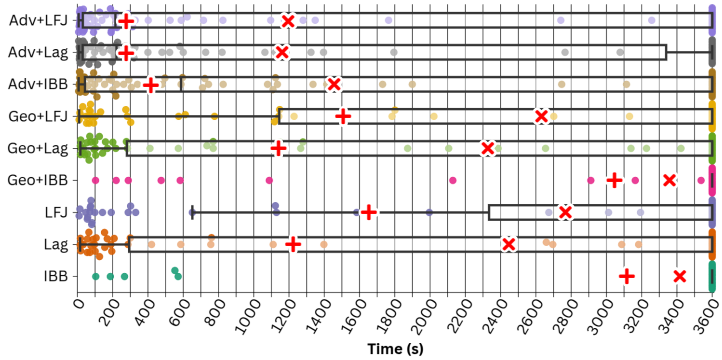


Fig. 7: The box plot of the running time of the problems in seconds, where at least one method is stopped by the time limit.

Figure 8 shows a box plot of the number of remaining boxes in the worklist for test cases where at least one method cannot solve the problem in one hour. Of the methods considered, the Adv+Lag and Adv+LFJ methods consistently result in the fewest number of unprocessed boxes.

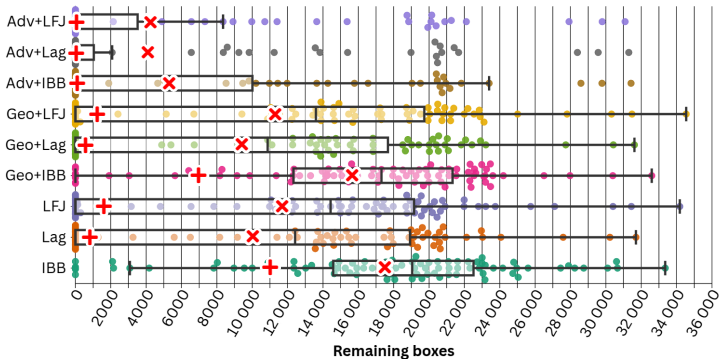


Fig. 8: The box plot of the remaining boxes of the all test cases, where at least one method reaches the time limit.

Figure 9 presents the percentage of solved problems for each method within the given time shown on the horizontal axis. Methods with the AdvTest demonstrate the best overall performance, achieving the highest success rate. It is clear that including this test consistently improves the problem-solving rate. This highlights the significant contribution of the AdvTest to the overall effectiveness of the methods. Between the Adv+LFJ and Adv+Lag methods, the latter is slightly better. However, in certain cases, the Adv+LFJ method outperforms the Adv+Lag method.

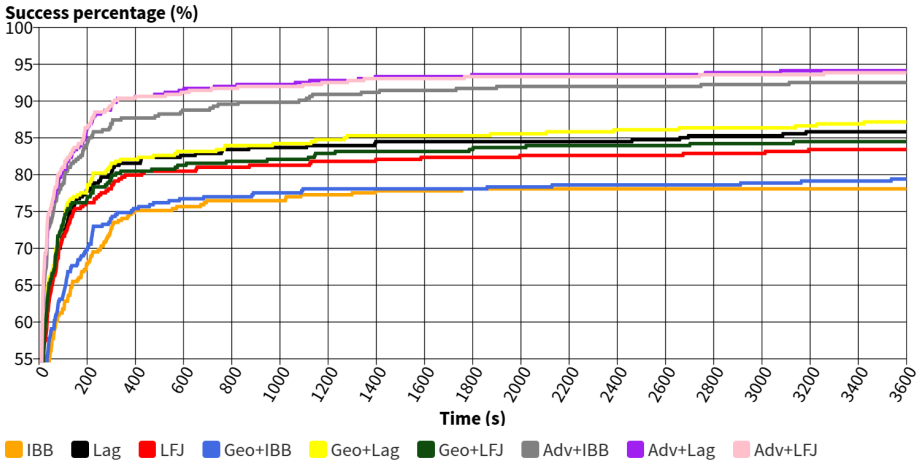


Fig. 9: The percentage of solved problems for each method within the given time.

Table 5 shows the average values of the main efficiency measures for the nine methods, where the best performing method for each metric is highlighted in bold. The first two measures show the arithmetic mean (Time (AM)) and the shifted geometric mean (Time (SGM)) of the calculation time. The Adv+Lag method is most effective for the arithmetic mean, while the Adv+LFJ method is most effective for the geometric mean. As shown in Figure 9, the two methods are very similar. Notably, the methods with the AdvTest are approximately 2.0-2.3 times faster than the methods with the basic Geometrical Test and 2.1-2.4 times faster than those without any Geometrical Test.

Table 5: The main efficiency measures for all test cases.

Methods	IBB	Lag	LFJ	Geo+IBB	Geo+Lag	Geo+LFJ	Adv+IBB	Adv+Lag	Adv+LFJ
Time (AM)	856.4	601.0	676.6	828.7	567.0	637.6	355.1	**286.5**	294.2
Time (SGM)	84.4	52.6	57.5	77.9	48.7	52.6	31.9	28.2	**27.8**
No. Iter.	5 543	3 950	4 216	5 198	3 657	3 899	2 425	**1 968**	1 996
Iter. Time	0.155	0.152	0.160	0.159	0.155	0.164	**0.146**	**0.146**	0.147
Rem. Box.	4 082	2 341	2 725	3 651	2 193	2 635	1 238	**954**	991
Max. WList	4 697	3 031	3 338	4 331	2 894	3 280	1 858	**1 560**	1 590
No. Opt. Test	–	3 493	3 753	4 780	3 260	3 509	2 229	**1 780**	1 807
Opt. (%)	–	15.8%	27.7%	14.8%	33.9%	43.1%	58.3%	64.7%	**66.0%**

The next two measures are No. Iter., the average number of iterations (i.e. enumerated boxes) and Iter. Time, the average time spent on an iteration (i.e. on a box). Adv+Lag achieves the best performance for both metrics. This suggests that the AdvTest effectively discards nonoptimal boxes. Interestingly, methods with the AdvTest spend less time per box than those without any Geometrical Test. Furthermore, methods using the AdvTest require significantly fewer iterations overall because many boxes are efficiently pruned from the upper levels of the computation tree.

The following two metrics describe the number of remaining boxes, Rem. Box., and the maximum length of the work list, Max. WList. Again, the Adv+Lag method is the best for these metrics. Using the AdvTest, the methods significantly reduce the number of function evaluations and memory usage, which is crucial for higher-dimensional problems.

The last two metrics are about the optimality tests. These metrics collectively refer to the Geometrical, Lag, and LFJ tests, as well as their combinations. The metrics show the average number of executed optimality tests (No. Opt. Test) and their success rate (Opt. (%)), defined as the percentage of skip, discard, and reduce outcomes. The Adv+Lag method performs best, requiring the fewest optimality test. However, the Adv+LFJ method has a one percent higher success rate. As we can see, methods with the AdvTest can skip, discard, or reduce the boxes in most cases.

In general, Adv+Lag is the best method to solve the generated problems in all the evaluated aspects. It significantly reduces the required calculation time, as well as the other metrics listed in Table 5. However, the Adv+LFJ method yields very similar

results and performs slightly better in some cases, particularly in terms of the success rate of optimality tests. Methods with the AdvTest are, on average, approximately 2.2 times faster than the other methods listed. In addition to being faster, these methods require less memory and computational resources, making them highly efficient choices for this problem class. Appendix D provides detailed information on the number of function evaluations and the behaviour of additional efficiency measures on the different discarding tests. Appendix E gives further insight into the efficiency of the different parts of the AdvTest. Tables containing detailed running times for all tests, along with other useful data, are available in the GitLab repository [1].

6 Summary

We investigated the Interval Branch and Bound method for solving constrained nonlinear optimization problems, with a particular focus on the applicability of optimality conditions. We developed the Advanced Geometrical Test compared to the basic Geometrical Test published in [16]. To achieve this, we analyzed the Fritz-John (FJ) optimality conditions from a geometric perspective. We observed that gradient enclosures provide sufficient information to determine whether a given box does not contain the optimum or whether the optimality conditions would not be effective in reducing the box. Based on the vector space interpretation of the gradients, we identified additional verifiable criteria that can be easily tested to determine whether a box potentially contains an optimal solution. The Advanced Geometrical Test evaluates each theoretical result step by step, focusing on reducing computational overhead. During this development, we identified efficient one-, two-, and higher-dimensional checks. Furthermore, we separated the problem into two main types based on the number of active constraints present in a given box: boxes with one active constraint and boxes with multiple active constraints. These observations enabled more targeted and efficient analysis.

We analyzed the performance of the methods on the set of 374 problems to provide a comprehensive evaluation. Using the Advanced Geometrical Test, the success rate of the optimality test improved, and the running time of the methods was significantly reduced. The most effective method was the IBB method combined with the Advanced Geometrical Test and the Lagrange estimator method (Adv+Lag). This method outperformed others in many key aspects. However, the performance of the IBB method using the Advanced Geometrical Test and the Lagrange estimator + NFJ method (Adv+LFJ) was very close in most of the investigated metrics. More specifically, methods with the Advanced Geometrical Test were approximately 2.0 to 2.3 times faster than methods using the basic Geometrical Test, and 2.1 to 2.4 times faster than those without any Geometrical Test. The highest success rate achieved among all tested methods was 66.0%, confirming the effectiveness of combining optimality conditions with the Advanced Geometrical Test.

In future work, we plan to integrate constraint propagation to further reduce the search space by the inclusion of the gradients. We also aim to extend the current methodology to handle equality constraints, thereby broadening its applicability to a wider class of constrained nonlinear optimization problems. Furthermore, we want

to adapt the method to mixed-integer problems, allowing it to address optimization tasks involving both continuous and integer variables.

Appendix A Detailed description of the FJ methods

Lagrange estimator method: In this method, we use the first $n+1$ equations from (7) only for the variables $\boldsymbol{\mu}$. The reduced system is solved using the ILUD method, which requires a square system. In the overdetermined case, only the first $|B|+|C|+1$ equations are considered. In the solution by ILUD, if any of the obtained bounds for the Lagrange multipliers is negative, we discard the box. Additionally, in the overdetermined case, we check if the Lagrange multipliers satisfy all the remaining equations. If not, we discard the box. In the underdetermined case, we do not solve the system of equations.
Lagrange estimator + NFJ method: This approach combines the Lagrange estimator method and the Newton method for (7). First, we use the Lag method to obtain bounds for the Lagrange multipliers. If we cannot discard the box and the bounds obtained for the Lagrange multipliers are within the interval $[0, 1]$, then we apply one step of the Newton method to (7) using the IGS method. If the solution is an empty set, there is no optimizer in the examined box $\boldsymbol{x}$. The IGS method divides the components where the coefficient of the diagonal interval contains zero and replaces the box $\boldsymbol{x}$ with the subboxes. In the worst case, the box remains unchanged.

Appendix B Detailed description of the Advanced Discarding Tests for feasible boxes

Monotonicity Test: If the inclusion of the gradient of the objective function does not contain zero, then the function is monotone. Thus, the interior of the box cannot contain an optimizer, which could only be on the face of the box, where the gradient decreases. We can discard the box if the function is monotone and the face of the box does not intersect with the boundary of the initial box. Otherwise, we narrow the box to the intersection. The test returns the narrowed box and a flag indicating whether the box has been discarded, reduced, or unchanged.
Non-convexity Test: It checks if the non-convexity of the objective function can be proved in the box. We calculate the inclusion of the second-order derivative, the Hessian matrix. For non-convexity, it suffices that the function is non-convex in one dimension. If a diagonal element in the Hessian matrix is negative, then the function cannot be convex. If the box is not on the boundary in the non-convex dimension (i.e. there are no active boundary constraints), it is discarded. Otherwise, it is narrowed to the intersection of the corresponding face(s) and the boundary. If both the lower and upper bounds are active, two degenerate boxes are produced.
Newton Test: It tries to narrow the box around stationary points. We apply one step of the interval Newton method for the objective function, which means solving an interval linear system using the IGS method. If the solution intersected by the current box is an empty set, we discard the box. We use the extended interval division to avoid failing to divide by an interval containing zero [17]. In this case, we divide the

box into at most $k+1$ boxes if k elements contain zero in the diagonal of the matrix. Thus, we may return many boxes. Sometimes, we cannot reduce the size of the box due to overestimation. Thus, we leave it unchanged.

Appendix C Supplementary Strategies for the Geometrical Test

We investigated four other modifications of the Advanced Geometrical Test to increase performance. However, we could not reduce the running time significantly.

In the first attempt, in the case of only one active constraint, we tried to solve the NFJ instead of stopping the test (Algorithm 1 line 26), but the running time did not reduce. In the second attempt, we tried to solve the reduced NFJ with the inclusion of the Lagrange multiplier $\boldsymbol{\lambda}$ only if it was finite, and the inclusion of the gradient of the active constraint does not contain zero in the interior (Proposition 8). Reducing the NFJ involves deleting the column corresponding to $\boldsymbol{\mu}_0$ because there is only one active constraint, and the gradient of the active constraint does not contain zero in the interior, so $\boldsymbol{\mu}_0 = 1$. Thus, we also delete the row corresponding to the normalization of the Lagrange multipliers (the first equation of (7)) because there is only one Lagrange multiplier. Solving the reduced system of equations was more successful than in the first attempt. However, skipping the NFJ conditions saved even more running time in general.

In the case of two or more active constraints, we also attempted to pair two undetermined coordinates in the DiscardByTwoDim method (Algorithm 4). In this method, we selected two undetermined coordinates and first checked if all quadrants were included in these two coordinates. Thus, the conic hull was full, so we stopped. Otherwise, there exists at least one empty quadrant, then we rotated the inclusion of the gradients such that the third quadrant $(--)$ is empty. Then, we calculated the line slope of the gradients' inclusion of the active constraints $\mathbf{m}_{ij}^g$ and the line slope of the gradients of the objective function $\mathbf{m}_{ij}^f$ in the two selected coordinates. It was difficult to calculate the slope of the constraints, and we could not apply the formula we provided before directly. The slopes must be calculated for the second $(+-)$ and fourth $(-+)$ quadrants. Then, the line slope can be obtained by combining them, taking into account the quadrant for which it was calculated. If $\mathbf{m}_{ij}^g \cap \mathbf{m}_{ij}^f = \emptyset$, then we discarded the box. However, in many cases, the angle enclosed between the upper and lower bounds of $\mathbf{m}_{ij}^g$ was greater than $180°$, so the conic hull was full in the chosen coordinates. As there are many coordinate pairs and only a few proved to be useful, using this method increased the running time, and we discarded the box only in a few cases.

Furthermore, to improve the efficiency of the method, we created two decision tree models to determine whether to solve or skip solving the optimality conditions. These models can easily be used in Matlab as if-else statements. We used the first decision tree at the end of the one constraint case (replacing the line 26 in Algorithm 1) and the second decision tree at the end of the two or more constraint case. We identified 34 possible features, such as the average diameter of the box, the average diameter of the gradient of the objective function, the percentage of determined coordinates, etc. Furthermore, we trained the models using the generated test cases and implemented

the decision trees as if-else statements. We found that the best models, which were sufficient to categorize the classes, had five levels and five features. Although using these models slightly reduced the running time of the methods in general, computing the features was computationally too intensive, and predicting the class was sometimes slower than directly calculating the optimality conditions.

Appendix D Additional efficiency measures

Table D1 shows the average values of the additional efficiency measures for the nine methods. The best performing method for each metric is highlighted in bold.

The first three rows compare the weighted number of function evaluations (WFE). WFE is calculated by WFE $= F_{eval} + k \cdot G_{eval} + \frac{k \cdot (k+1)}{2} \cdot H_{eval}$, where k is the number of dependent variables, F_{eval}, G_{eval}, and H_{eval} are the number of function, gradient and Hessian evaluations, respectively. The WFE of the objective function (Obj. WFE), the WFE of the constraints (Const. WFE), and the WFE of all functions (All WFE) yield interesting results. Among all the methods, the Adv+Lag method is the best for these metrics. The AdvTest significantly reduces the number of function evaluations, which is crucial for higher-dimensional problems. Interestingly, when comparing the Const. WFE results between the IBB method with the basic Geometrical Test and those without any Geometrical Test, the latter group performs better. Nevertheless, using the AdvTest leads to a much more pronounced reduction in overall function evaluations, confirming its effectiveness in improving computational efficiency.

Table D1: The additional efficiency measures for all test cases.

Methods	IBB	Lag	LFJ	Geo+IBB	Geo+Lag	Geo+LFJ	Adv+IBB	Adv+Lag	Adv+LFJ
Obj. WFE	84 485	58 733	73 883	78 516	54 339	66 031	36 467	**29 793**	31 729
Const. WFE	33 369	48 826	62 487	58 685	42 641	51 061	34 996	**26 525**	27 511
All WFE	117 855	107 559	136 370	137 201	96 980	117 092	71 463	**56 318**	59 240
Nat. Cut. Test	26 787	16 735	18 012	24 520	15 296	16 507	9 369	**7 507**	7 580
Nat. Cut. (%)	8.0%	8.5%	8.6%	6.8%	7.1%	6.2%	6.1%	6.0%	5.9%
Cent. Cut. Test	24 636	15 312	16 458	22 857	14 216	15 477	15 884	12 092	**11 888**
Cent. Cut. (%)	2.0%	1.7%	1.8%	1.7%	1.4%	1.5%	2.1%	1.5%	1.5%
List Cut. Test	**39**	**39**	**39**	**39**	**39**	**39**	40	40	41
List Cut. (%)	53.6%	52.5%	50.6%	52.6%	51.9%	50.0%	49.0%	48.3%	47.3%
Feas. Test	18 380	10 876	11 721	17 044	10 123	11 111	5 995	**4 756**	4 809
Infeas. (%)	28.3%	20.9%	22.2%	28.7%	20.9%	22.5%	13.0%	10.6%	10.1%
Feas. (%)	2.4%	2.5%	2.8%	1.8%	1.7%	2.0%	1.6%	1.9%	2.0%
Undeter. (%)	69.2%	76.6%	75.0%	69.5%	77.4%	75.5%	85.4%	87.5%	87.9%
Monoton. Test	280	244	254	221	194	192	**161**	**161**	162
Monoton. (%)	60.4%	54.6%	55.8%	49.7%	42.9%	42.1%	31.0%	30.9%	31.6%

The following groups show the number of discarding tests and their corresponding success rates. The first three groups focus on the natural inclusion CutOff Test (Nat. Cut. Test), centred inclusion CutOff Test(Cent. Cut. Test) and list removal CutOff Test (List Cut. Test). The Adv+Lag and Adv+LFJ methods require the fewest

natural and centred inclusion Cutoff Tests, indicating greater efficiency in early-stage pruning. In contrast, the number of list removal Cutoff Tests remains relatively consistent across all methods. This behaviour is strongly related to the number of iterations. Methods with the AdvTest rely significantly on optimality tests for pruning, reducing the effectiveness of Cutoff Tests and their importance.

The next group focuses on the Feasibility Test, which includes the number of tests applied (Feas. Test) and the percentage of boxes classified as infeasible (Infeas. (%)), feasible (Feas. (%)), and undetermined (Undeter. (%)). Methods with the AdvTest require the fewest Feasibility Tests. However, they produce a higher proportion of undetermined boxes. This is because the AdvTest targets undetermined boxes earlier in the process. Additionally, using optimality tests significantly reduces the number of Feasibility Tests needed because many boxes are discarded earlier based on optimality criteria.

The final group shows the number of Monotonicity Test (Monoton. Test) along with their corresponding success rates. Methods with the AdvTest perform the fewest monotonicity checks and have lower success rates than other methods. In contrast, methods without the Geometrical Test perform significantly more monotonicity evaluations. This is because subproblems are usually divided until a feasible box is obtained, at which point the Monotonicity Test is attempted. Consequently, many of these attempts are unsuccessful, resulting in a higher overall number of monotonicity checks. This behaviour highlights the efficiency advantage of early-stage pruning in methods with any Geometrical Test. The Non-convexity Test and the Newton Test behave very similarly across all methods, hence those are not listed in the table.

Appendix E Detailed effectiveness of the Geometrical Test

Table E2 compares the basic Geometrical Test and the AdvTest, with or without Lag and LFJ, by the average number of tests (No. Geo. Test) and the average percentage of decisions made. In line Geo. Solve FJ (%), we report the percentage of cases in which optimality conditions were used, while Geo. Skip FJ (%) indicates the percentage of cases where solving FJ conditions was skipped. Geo. Disc. Box (%) refers to the percentage of instances in which the box was discarded. For the Geo+IBB and Adv+IBB methods, the reported Geo. Solve FJ and Geo. Skip FJ values are the recommendations of the Geometrical tests, as no Lag or LFJ were solved. Methods with the AdvTest require the fewest average number of tests used. This is due to the test's ability to discard nonoptimal subboxes at an earlier stage of the method. The test significantly reduces the percentage of decisions where the optimality condition is solved, while increasing the proportion of decisions that either skip the FJ conditions or discard the box. These results highlight the effectiveness of the AdvTest in improving both efficiency and pruning capability. The Adv+Lag method is the most effective, as evidenced by its high success rate in both efficiency indicators, Geo. Skip FJ (%) and Geo. Disc. Box (%). However, the Adv+LFJ method exhibits a similar behaviour, although its results are slightly less favourable compared to Adv+Lag.

Table E2: Detailed effectiveness of the geometrical tests

Methods	Geo+IBB	Geo+Lag	Geo+LFJ	Adv+IBB	Adv+Lag	Adv+LFJ
No. Geo. Test	4 780	3 260	3 509	2 229	1 780	1 807
Geo. Solve FJ (%)	85.2%	79.7%	81.9%	41.7%	36.3%	37.5%
Geo. Skip FJ (%)	10.1%	14.9%	13.8%	32.0%	40.1%	39.4%
Geo. Disc. Box (%)	4.7%	5.4%	4.3%	26.3%	23.6%	23.1%

Table E3: Performance of the tests in the Advanced Geometrical Test.

Methods	Adv+IBB	Adv+Lag	Adv+LFJ
No. Full **CF** Test, line 1	2 229	1 780	1 807
Full **CF** (%)	29.7%	37.2%	36.5%
No. Geo. Centred Form Cut. Test, line 5	1 567	1 119	1 147
Geo. Centred Form Cut. (%)	3.3%	3.3%	3.2%
No. Full **CH** Test, line 11	1 515	1 082	1 110
Full **CH** (%)	0.000005%	0.000007%	0.000007%
No. Geo. Monotonicity Test, line 16	5.6	4.1	4.1
Geo. Monotonicity (%)	95.1%	93.4%	93.4%
No. Sign Test, line 18	1 509	1 077	1 106
Sign (%)	13.5%	11.5%	10.6%
No. Independent Test, Algorithm 2, line 16	1 305	953	988
Independent Discard (%)	20.6%	21.3%	20.9%
Independent Solve Reduced FJ (%)	65.0%	59.3%	56.2%
No. All Orthant Used Test, line 21	363	305	343
All Orthant Used (%)	0.075%	0.089%	0.080%
No. One Const. Lag. Encl. Test, line 24	81	81	80
One Const. Lag. Encl. (%)	37.7%	36.3%	35.9%
No. More Const. Lag. Encl. Test, line 28	282	224	262
More Const. Lag. Encl. (%)	5.4%	6.2%	5.6%
No. Discard by Two Coord. Test, line 30	267	210	248
Discard by Two Coord. (%)	4.4%	4.3%	3.7%

Table E3 presents the average number of checks performed during the AdvTest, along with their respective success rates. These checks correspond to the steps in Algorithm 1, listed in their order of execution and referenced by their pseudocode line numbers. As we can see, the most skipped decision is made at the very beginning of the method, and very few check the full **CH** and all orthants used. Monotonicity is quite effective, although it can only be applied a couple of times. The Sign Test can eliminate around 100 boxes. However, what is really effective is the Independent Test, where we check Proposition 4. More than ten percent of all Adv Tested boxes were rejected there. The test that calculates the enclosure of the Lagrange multiplier with one active constraint is quite effective. The method becomes less efficient when the same is done for the enclosure of the gradients. Finally, the method of discarding by two coordinates could only remove a few boxes.

Declarations

Author contribution

Mihály Gencsi: Conceptualization, Methodology, Software, Visualization, Investigation, Validation, Writing - Original Draft.
Boglárka G.-Tóth: Validation, Funding acquisition, Formal analysis, Investigation, Writing - Review & Editing, Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data and Code availability

The source code and the benchmark instances used in this study are available at https://gitlab.com/gencsimiska27/the-ibb-with-geo-and-fj-test-intlab.

Funding

This research was funded. Specific grant details and project numbers were omitted to preserve author anonymity during the review process. These details will be added once the manuscript is accepted.

References

[1] Araya, I., Reyes, V.: Interval branch-and-bound algorithms for optimization and constraint satisfaction: a survey and prospects. Journal of Global Optimization **65**, 837–866 (2016) https://doi.org/10.1007/s10898-015-0390-4

[2] Pál, L., Csendes, T.: INTLAB implementation of an interval global optimization algorithm. Optimization Methods and Software **24**(4-5), 749–759 (2009) https://doi.org/10.1080/10556780902753395

[3] Kearfott, R.B.: An interval branch and bound algorithm for bound constrained optimization problems. Journal of Global Optimization **2**(3), 259–280 (1992) https://doi.org/10.1007/BF00171829

[4] Ninin, J.: Global optimization based on contractor programming: An overview of the IBEX library. In: Kotsireas, I.S., Rump, S.M., Yap, C.K. (eds.) Mathematical Aspects of Computer and Information Sciences, pp. 555–559. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-32859-1_47

[5] Carrizosa, E., Messine, F.: An interval branch and bound method for global robust optimization. Journal of Global Optimization **80**, 507–522 (2021) https://doi.org/10.1007/s10898-021-01010-5

[6] Hansen, E.R., Walster, G.W.: Nonlinear equations and optimization. Computers & Mathematics with Applications **25**(10), 125–145 (1993) https://doi.org/10.1016/0898-1221(93)90288-7

[7] Kearfott, R.B.: Rigorous Global Search: Continuous Problems. Springer, Boston, MA (1996). https://doi.org/10.1007/978-1-4757-2495-0

[8] Mangasarian, O.L., Fromovitz, S.: The Fritz John necessary optimality conditions in the presence of equality and inequality constraints. Journal of Mathematical Analysis and Applications **17**(1), 37–47 (1967) https://doi.org/10.1016/0022-247X(67)90163-1

[9] Shary, S.P.: Interval Gauss-Seidel method for generalized solution sets to interval linear systems. Reliable Computing **7**(2), 141–155 (2001) https://doi.org/10.1023/A:1011422215157

[10] Goldsztejn, A., Chabert, G.: A generalized interval LU decomposition for the solution of interval linear systems. In: Numerical Methods and Applications, pp. 312–319. Springer, Berlin, Heidelberg (2007). https://doi.org/10.1007/978-3-540-70942-8_37

[11] Hansen, E., Walster, G.W.: Global Optimization Using Interval Analysis: Revised And Expanded. CRC Press, Boca Raton (2003). https://doi.org/10.1201/9780203026922

[12] Kearfott, R.B.: Preconditioners for the interval Gauss-Seidel method. SIAM Journal on Numerical Analysis **27**(3), 804–822 (1990) https://doi.org/10.1137/0727047

[13] Hansen, E.R., Walster, G.W.: Bounds for Lagrange multipliers and optimal points. Computers & Mathematics with Applications **25**(10), 59–69 (1993) https://doi.org/10.1016/0898-1221(93)90282-Z

[14] Jaulin, L., Kieffer, M., Didrit, O., Walter, E.: Applied Interval Analysis with Examples in Parameter and State Estimation, Robust Control and Robotics. Springer, London (2001). https://doi.org/10.1007/978-1-4471-0249-6

[15] Gencsi, M., G.-Tóth, B.: The Fritz-John condition system in interval branch and bound method. Annales Mathematicae et Informaticae, 56–68 (2023) https://doi.org/10.33039/ami.2023.08.005

[16] Gencsi, M., G.-Tóth, B.: Efficient use of optimality conditions in interval branch and bound methods. EURO Journal on Computational Optimization **13**, 100108 (2025) https://doi.org/10.1016/j.ejco.2025.100108

[17] Kulisch, U.W.: Complete interval arithmetic and its implementation on the

computer. In: Cuyt, A., Krämer, W., Luther, W., Markstein, P. (eds.) Numerical Validation in Current Hardware Architectures, pp. 7–26. Springer, Berlin, Heidelberg (2009). https://doi.org/10.1007/978-3-642-01591-5_2

[18] Rump, S.M.: INTLAB - INTerval LABoratory. In: Csendes, T. (ed.) Developments in Reliable Computing, pp. 77–104. Kluwer Academic Publishers, Dordrecht (1999). http://www.tuhh.de/ti3/rump/

[19] Griewank, A., Corliss, G.F.: Automatic Differentiation of Algorithms: Theory, Implementation, and Application. Society for Industrial and Applied Mathematics (SIAM), Philadelphia (1992)

[20] Andreani, R., Martinez, J.M., Schuverdt, M.L.: On the relation between constant positive linear dependence condition and quasinormality constraint qualification. Journal of Optimization Theory and Applications **125**(2), 473–483 (2005) https://doi.org/10.1007/s10957-004-1861-9

[21] Mangasarian, O.L.: Nonlinear Programming. Classics in Applied Mathematics. Society for Industrial and Applied Mathematics (SIAM), ??? (1994). https://doi.org/10.1137/1.9781611971255

[22] The MathWorks Inc.: MATLAB Version 9.8.0 (R2020a). Natick, Massachusetts, United States (2020). https://www.mathworks.com

[23] Csendes, T.: Numerical experiences with a new generalized subinterval selection criterion for interval global optimization. Reliable Computing **9**(2), 109–125 (2002) https://doi.org/10.1023/A:1023086201037