Constructing Tight Quadratic Relaxations for Global Optimization: I. Outer-Approximating Twice-Differentiable Convex Functions

William R. Strahl[1,2], Arvind U. Raghunathan[3], Nikolaos V. Sahinidis[2,4], and Chrysanthos E. Gounaris[1,2*]

[1]Department of Chemical Engineering, Carnegie Mellon University, Pittsburgh, PA, 15213, USA

[2]Center for Advanced Process Decision-making, Carnegie Mellon University, Pittsburgh, PA, 15213, USA

[3]Mitsubishi Electric Research Labs, Cambridge, MA, 02139, USA

[4]H. Milton Stewart School of Industrial & Systems Engineering and School of Chemical & Biomolecular Engineering, Georgia Institute of Technology, Atlanta, GA, 30332, USA

Abstract

When computing bounds, spatial branch-and-bound algorithms often linearly outer approximate convex relaxations for non-convex expressions in order to capitalize on the efficiency and robustness of linear programming solvers. Considering that linear outer approximations sacrifice accuracy when approximating highly nonlinear functions and recognizing the recent advancements in the efficiency and robustness of available methods to solve optimization problems with quadratic objectives and constraints, we contemplate here the construction of quadratic outer approximations of twice-differentiable convex functions for use in deterministic global optimization. To this end, we present a novel cutting-plane algorithm that determines the tightest scaling parameter, α, in the second-order Taylor series approximation quadratic underestimator proposed by Su et al. [2018]. We use a representative set of convex functions extracted from optimization benchmark libraries to showcase–qualitatively and quantitatively–the tightness of the constructed quadratic underestimators and to demonstrate the overall computational efficiency of our algorithm. Furthermore, we extend our construction procedure to generate even tighter quadratic underestimators by allowing overestimation in infeasible polyhedral regions of optimization problems, as informed by the latter's linear constraints.

*Corresponding author: gounaris@cmu.edu

Keywords: deterministic global optimization, outer approximation, quadratic underestimation, cutting-plane algorithm

1 Introduction

As evidenced by the vast diversity of application areas present in the literature, deterministic global optimization (DGO) algorithms are utilized to solve problems to guaranteed global optimality in many different fields, including biomedical and biological engineering, computational chemistry, computational geometry, finance, process networks, and others. A brief, yet diverse subset of specific applications where deterministic DGO algorithms have been applied includes bioreactors [Misener et al., 2014], protein design [Klepeis et al., 2004], phase equilibrium [Pereira et al., 2010], membrane-based separation systems [Gounaris et al., 2013], investment portfolio selection [Rios and Sahinidis, 2010], and crude oil scheduling [Mouret et al., 2011]. Recent applications include optimizing heat integration in flowsheets [Fahr et al., 2022], parameter estimation problems in the chemical industry [Sass et al., 2023], and cost minimization of water distribution networks [Cassiolato et al., 2023], to name but a few. Indeed, advancing DGO techniques produces enhanced algorithms that can be utilized in a plethora of problems encountered across a vast array of research fields.

Many DGO algorithms extensively utilize supporting hyperplanes (i.e., first-order Taylor series approximations) for nonlinear convex functions, which have guarantees from convexity theory to underestimate convex functions over the entire domain (e.g., [Tawarmalani and Sahinidis, 2005]). Most notably, spatial branch-and-bound algorithms, which repeatedly construct and refine convex relaxations of the typically non-convex feasible space, often opt for linear outer-approximations of those relaxations at the interest of solving them faster and more reliably. Whereas they supply valid underestimators necessary for rigorous guarantees of optimality and are very inexpensive to construct, linear outer-approximators can poorly approximate highly nonlinear functions. More broadly, directly embedding the nonlinear convex functions into any intermediate computation of a DGO algorithm immediately elevates the difficulty of the problem to solving general convex optimization problems, for which efficient methods exist, but are computationally less efficient and/or less robust compared to linear programming (LP) algorithms. Moreover, recent advances in the algorithms solving problems with quadratic objectives and/or constraints [Mittelmann, 2023] sug-

gests an intermediate option: using a nonlinear quadratic function as an outer approximator to more accurately approximate nonlinear functions, while at the same time maintaining a lower level of difficulty than general nonlinear convex programming. This paradigm has been applied, for example, in a decomposition algorithm for convex Mixed Integer Nonlinear Programming (MINLP) problems, where an Outer Approximation (OA) scheme generates quadratic cuts defining a Mixed Integer Quadratically Constrained Quadratic Program (MIQCQP) relaxed master problem [Su et al., 2018]. Furthermore, any algorithm or technique that linearly outer approximates convex relaxations could instead consider quadratic outer approximations, which provide an intermediate alternative within the complexity gap between nonlinear convex programming and linear programming, allowing for enhanced accuracy of relaxation solutions at a greater computational cost. In this context, another possible application lies in optimization-based bounds tightening (OBBT) [Puranik and Sahinidis, 2017, Bynum et al., 2021], which tightens bounds on variables by minimizing and maximizing variables subject to a relaxation of the problem. Rather than utilizing a linear outer approximation of a convex relaxation and solving LPs, tighter bounds could be achieved by solving the OBBT problems over a quadratic outer approximation of the convex relaxation. The broadly applicable principle of replacing linear outer approximations with quadratic outer approximations in DGO algorithms motivates this work to provide a methodology for the construction of tight quadratic outer approximators for twice-differentiable convex functions. We emphasize that our contributions are not restricted to any particular algorithm or technique, but apply generally to all contexts where outer approximation is relevant for rigorous optimality guarantees.

Quadratic outer approximators (a.k.a."quadratic underestimators"; terms used interchangeably in this work) have been proposed in the literature and utilized in solving optimization problems. Buchheim and Trieu [2013] solve general convex integer programs by using a point-wise maximum quadratic surrogate for nonlinear functions. In their distinct approach the authors construct quadratic outer approximators for a specific class of functions where the Hessian, Q, of a second-order Taylor series approximation is selected to satisfy $\nabla^2 f(x) \succcurlyeq Q$ for all $x \in \mathbb{R}^n$. While identifying a Q that satisfies this condition a priori is not in general applicable to nonlinear functions, the underestimator possesses the property that the same Hessian, Q, can be reused for any point of construction in the domain without modification; once identified, construction is cheap and tighter than linear, but not as tight as possible. The authors acknowledge the difficulty of finding a suit-

able Q for general nonlinear functions and suggest determining Q for additional specific classes of functions as future work [Buchheim and Trieu, 2013].

Su et al. [2018] construct quadratic outer approximators for use in outer approximation algorithms for convex MINLP problems, where the cuts added to the master problem are convex quadratic instead of linear. Rather than selecting the same Hessian for the second-order term in a Taylor series approximation, the authors propose modifying the Hessian uniquely for each point of construction, thereby achieving tighter quadratic outer approximators. Their modified Taylor series approximation at a point of construction x_0 takes the form $f(x_0) + \nabla f(x_0)(x - x_0) + \frac{1}{2}(x - x_0)^\top \alpha \nabla^2 f(x_0)(x - x_0)$, where the scalar α modifies the second-order term to achieve underestimation over a pre-specified box domain $\mathcal{B}$. Acknowledging the difficulty of solving the non-convex optimization problem (1) to determine the tightest possible value for α, the authors observe that, in the case when the gradient of the objective function in (1) is coordinate-wise monotonic (i.e., each component of the gradient is monotonic over the feasible space), then the tightest possible choice for α (and the solution to (1)) lies at a vertex of the box domain. However, while providing a procedure for identifying the tightest α for functions that satisfy this coordinate-wise monotonicity property, Su et al. [2018] leave open the problem of determining the tightest α for general nonlinear convex functions.

$$\alpha = \min_{x \in \mathcal{B} \setminus \{x_0\}} \frac{f(x) - \big(f(x_0) + \nabla f(x_0)(x - x_0)\big)}{\frac{1}{2}(x - x_0)^\top \nabla^2 f(x_0)(x - x_0)} \tag{1}$$

Olama et al. [2023] consider an outer approximation algorithm that employs a quadratic underestimator of the form $f(x_0) + \nabla f(x_0)(x - x_0) + \frac{m}{2}(x - x_0)^\top (x - x_0)$, where f is strongly convex given a suitable choice of m. Closely resembling the approach proposed in Buchheim and Trieu [2013], this methodology provides a quadratic underestimator that applies to any point of construction for a function, but also possesses the drawback that computing m is only straightforward for a restricted class of functions (e.g., convex quadratic functions in problems of learning and control). Integrated into the outer approximation algorithm, the authors claim that an advantage of their quadratic outer approximator over the one proposed by Su et al. [2018] is that, by utilizing the strong convexity parameter, the Hessian of the quadratic constraint included in the outer approximation master problem is a diagonal matrix, and thereby less dense and more efficient to solve by available solvers [Olama et al., 2023]. Finally, the authors assert that integrating the quadratic outer approximation of the objective function of the learning and control type of problems studied

(e.g., classification, optimal and predictive control, regression) computes tighter lower bounds for the outer approximation algorithm and facilitates faster convergence.

Recent work by Streeter and Dillon [2022] develops the theory to construct polynomial under- and over-estimators of functions using so-called Taylor polynomial enclosures. Additionally, the authors present a detailed software implementation for their numerical procedure, named "Auto-Bound", which can construct quadratic under- and over-estimators for general non-convex functions, but provides no guarantee on the convexity of the resulting estimators. Furthermore, Streeter and Dillon [2023] show that the analytic expressions, rather than numerical procedures, can generate the *tightest* estimators for univariate functions composed of functions with a monotonic second derivative (in the case of using quadratics as estimators). Considered in the context of the quadratic underestimator proposed in Su et al. [2018], for univariate functions composed of functions with a monotonic second derivative, these analytic expressions identify *a priori* which vertex is the solution of (1) and determine the tightest scaling parameter α.

Despite all this recent research activity, we highlight that, to the best of our knowledge, the literature lacks a procedure to construct tight *convex* quadratic underestimators of twice-differentiable convex functions in general. To this end, we provide the following distinct contributions:

- We set forth a cutting-plane algorithm that numerically determines the tightest value of α in the underestimator proposed by Su et al. [2018] for general twice-differentiable convex functions of low dimensions.
- We provide qualitative and quantitative evidence of the quality of the quadratic underestimators and demonstrate the efficiency of the cutting-plane algorithm on a set of representative functions extracted from optimization benchmark libraries.
- We extend the cutting-plane algorithm to generate even tighter quadratic underestimators of functions present in optimization problems by allowing overestimation in infeasible regions, as informed by linear constraints in those problems.

The rest of the paper is organized as follows. In Section 2, we present our approach for constructing tight quadratic underestimators of general twice-differentiable convex functions, while in Section 3 we present results on a library of convex functions. In Section 4, we show how to construct even tighter underestimators in the context of optimization problems with linear constraints,

illustrating this extension with a representative example. Finally, in Section 5, we present some conclusions from our work.

2 Methodology

For underestimating a twice-differentiable convex function, $f : \mathbb{R}^n \to \mathbb{R}$, let $\mathcal{B} := \{\boldsymbol{x} \in \mathbb{R}^n : x_i^L \leq x_i \leq x_i^U \ \forall i = 1, \ldots, n\}$ denote the box defined by given bounds x_i^L and x_i^U on the components of $\boldsymbol{x}$, and let $\boldsymbol{x}_0 \in \mathcal{B}$ be a point at which to construct a quadratic underestimator,

$$q(\boldsymbol{x}; \alpha, \boldsymbol{x}_0) := f(\boldsymbol{x}_0) + \nabla f(\boldsymbol{x}_0)(\boldsymbol{x} - \boldsymbol{x}_0) + \frac{1}{2}(\boldsymbol{x} - \boldsymbol{x}_0)^\top \alpha \nabla^2 f(\boldsymbol{x}_0)(\boldsymbol{x} - \boldsymbol{x}_0), \tag{2}$$

where α is a scaling parameter selected to satisfy $q(\boldsymbol{x}; \alpha, \boldsymbol{x}_0) \leq f(\boldsymbol{x})$ for all $\boldsymbol{x} \in \mathcal{B}$. We note that herein and in the remainder of the paper, we denote vectors in bold.

Adopting the second-order Taylor series approximation at $\boldsymbol{x}_0$ for the quadratic underestimator (2), as presented in Su et al. [2018], we seek to solve the optimization problem (3). Here, we note that α is necessarily bounded from above by 1, or else the Taylor series approximation would overestimate in the local vicinity of the point of construction. Additionally, we highlight that, in order to be properly defined, the quadratic (2) requires local twice-differentiability at the point of construction, $\boldsymbol{x}_0$, itself. While in this exposition we define function f to be twice-differentiable across its full domain, allowing us to readily compute underestimators using a variety of points of construction, we highlight that the main idea of our work can be extended to once-differentiable functions, as long as care is taken to consider only locally twice-differentiable points of construction.

$$\begin{aligned} \max_{\alpha \in [0,1]} \quad & \alpha \\ \text{s.t.} \quad & \min_{\boldsymbol{x} \in \mathcal{B}} \{f(\boldsymbol{x}) - q(\boldsymbol{x}, \alpha; \boldsymbol{x}_0)\} \geq 0 \end{aligned} \tag{3}$$

For a fixed value of α, the minimization problem defining the feasible set of (3) is a difference-of-convex (d.c.) function minimization problem that, by utilizing a partial epigraph reformulation, can be cast as a concave minimization problem over a convex feasible set, as displayed in (4).

$$\begin{aligned} \min_{\boldsymbol{x} \in \mathcal{B}, t \in \mathbb{R}} \quad & t - q(\boldsymbol{x}; \alpha, \boldsymbol{x}_0) \\ \text{s.t.} \quad & f(\boldsymbol{x}) - t \leq 0 \end{aligned} \tag{4}$$

Various approaches exist in the literature for solving (4), including outer approximation, polyhedral annexation, successive underestimation, and branch-and-bound [Horst and Pardalos, 2013].

Polyhedral annexation and successive underestimation methods require a polyhedral feasible region, which is not applicable to our specific concave minimization problem inasmuch as the constraint in (4) in nonlinear in general. Furthermore, while branch-and-bound can in principle solve problems with convex non-polyhedral feasible sets, it suffers intensely from the curse of dimensionality due to the requirement to branch spatially on continuous domains. Hence, we chose to implement an outer approximation scheme utilizing a cutting-plane algorithm to solve (4). Exploiting the monotonicity of (2) as a function of α, as detailed in Observation 1, we maintain a set of vertices corresponding to an outer approximation of the convex feasible region of (4), which we evaluate against the objective function of (4) to compute a bound on the overestimation of the function by the quadratic for a particular value of α. We then iteratively refine our outer approximation vertex set using a cutting-plane algorithm to separate vertices until the bound computed by (4) is within some specified tolerance, ε. To guarantee convergence of the algorithm and simultaneously determine the tightest parameter α, i.e., the solution to (3), we initialize the cutting-plane algorithm with $\alpha = 1$ and decrement the value of α at each enumerated vertex of the cutting-plane algorithm if (2) overestimates the function. The cutting-plane algorithm is presented as a diagram in Figure 1, while detailed explanations of each step are provided in Section 2.1.

Observation 1. *The quadratic underestimator in (2) monotonically decreases as α decreases; that is, $q(\boldsymbol{x}; \alpha_1, \boldsymbol{x}_0) \leq q(\boldsymbol{x}; \alpha_2, \boldsymbol{x}_0)$ for all $\alpha_1 < \alpha_2$.*

Proof. By construction, $\nabla^2 f(\boldsymbol{x}_0) \succcurlyeq 0$, which implies that $\frac{1}{2}(\boldsymbol{x} - \boldsymbol{x}_0)^\top \nabla^2 f(\boldsymbol{x}_0)(\boldsymbol{x} - \boldsymbol{x}_0) \geq 0$ for all $\boldsymbol{x} \in \mathbb{R}^n$. Let two values for parameter α, namely $\alpha_1 \in [0, 1)$ and $\alpha_2 \in (0, 1]$ such that $\alpha_1 < \alpha_2$. We thus have:

$$\begin{aligned} \alpha_1 < \alpha_2 \quad &\implies \tfrac{\alpha_1}{2}(\boldsymbol{x} - \boldsymbol{x}_0)^\top \nabla^2 f(\boldsymbol{x}_0)(\boldsymbol{x} - \boldsymbol{x}_0) \leq \tfrac{\alpha_2}{2}(\boldsymbol{x} - \boldsymbol{x}_0)^\top \nabla^2 f(\boldsymbol{x}_0)(\boldsymbol{x} - \boldsymbol{x}_0) \\ &\iff q(\boldsymbol{x}; \alpha_1, \boldsymbol{x}_0) \leq q(\boldsymbol{x}; \alpha_2, \boldsymbol{x}_0), \end{aligned}$$

noting that equality holds only at points $\boldsymbol{x}^* \in \mathbb{R}^n$ for which $\frac{1}{2}(\boldsymbol{x}^* - \boldsymbol{x}_0)^\top \nabla^2 f(\boldsymbol{x}_0)(\boldsymbol{x}^* - \boldsymbol{x}_0) = 0$. □

2.1 The Cutting-Plane Algorithm

Let $\mathcal{U}^{(k)}$ be the set of vertices at iteration k that outer approximates the epigraph of f, and let $\mathcal{A}^{(k)} \subseteq \mathcal{U}^{(k)}$ be the set of vertices to be evaluated against the objective in (4). Let $\mathcal{B}$ be the set of points in the continuous box domain defined by the bounds on variables $\boldsymbol{x}$. Let $\mathcal{V}$ be the 2^n vertices of $\mathcal{B}$, where n is the dimension of vector $\boldsymbol{x}$. Finally, let $g(\boldsymbol{x}, t) := f(\boldsymbol{x}) - t$.

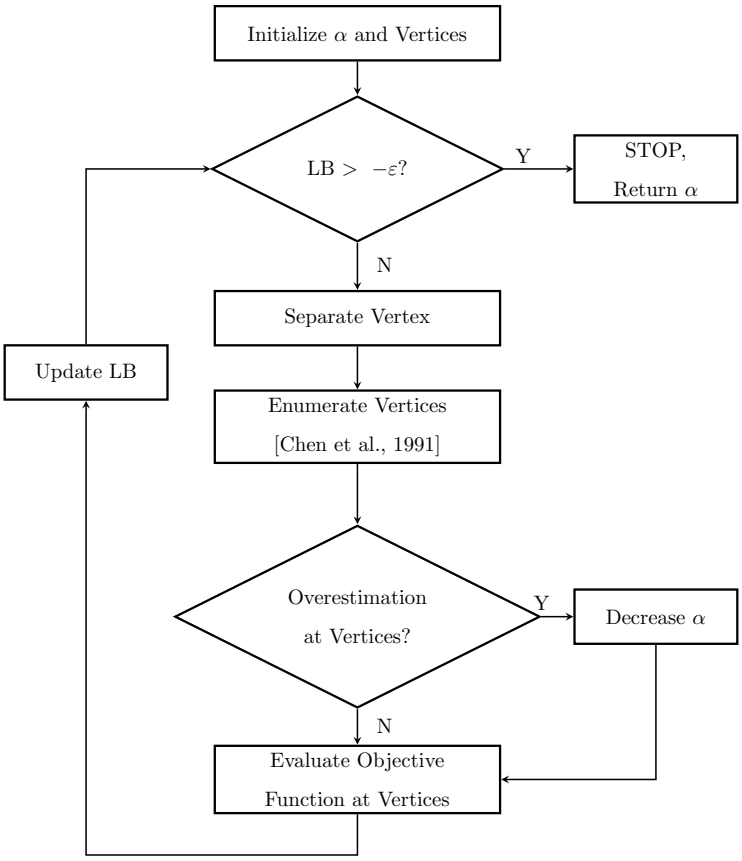


Figure 1. Diagram illustrating the cutting-plane algorithm.

1. *Initialize:* Choose tolerance ε. Set $k \leftarrow 0$,
   $\mathcal{A}^{(0)}, \mathcal{U}^{(0)} \leftarrow \big\{(\boldsymbol{x}, t) : \boldsymbol{x} \in \mathcal{V}, t \in \{t^L, t^U\}\big\}$, where $t^L := \min_{\boldsymbol{x} \in \mathcal{B}} f(\boldsymbol{x})$ and $t^U := \max_{\boldsymbol{v} \in \mathcal{V}} f(\boldsymbol{v})$,
   $\alpha^{(0)} \leftarrow \min\limits_{(\boldsymbol{x}_v, t_v) \in \mathcal{U}^{(0)}} \left\{ \frac{f(\boldsymbol{x}_v) - \big(f(\boldsymbol{x}_0) + \nabla f(\boldsymbol{x}_0)(\boldsymbol{x}_v - \boldsymbol{x}_0)\big)}{\frac{1}{2}(\boldsymbol{x}_v - \boldsymbol{x}_0)^\top \nabla^2 f(\boldsymbol{x}_0)(\boldsymbol{x}_v - \boldsymbol{x}_0)} \;:\; f(\boldsymbol{x}_v) - q(\boldsymbol{x}_v; 1, x_0) < -\varepsilon \right\}$.
   Select a point $(\boldsymbol{x}_p, t_p) \in \mathcal{B} \times [t^L, t^U]$ such that $f(\boldsymbol{x}_p) < t_p$. Initialize the vertex enumeration data structures as described by Chen et al. [1991].

2. *Check Convergence:* $LB = \min\limits_{(\boldsymbol{x}, t) \in \mathcal{A}^{(k)}} t - q(\boldsymbol{x}; \alpha^{(k)}, \boldsymbol{x}_0)$.
   If $LB > -\varepsilon$, then STOP, since $f(\boldsymbol{x}) - q(\boldsymbol{x}; \alpha^{(k)}, \boldsymbol{x}_0) \geq -\varepsilon$ for all $\boldsymbol{x} \in \mathcal{B}$, and return $\alpha^{(k)}$.

3. *Separate Vertex:* $\boldsymbol{u} \leftarrow \operatorname{argmin}\limits_{(\boldsymbol{x}, t) \in \mathcal{A}^{(k)}} t - q(\boldsymbol{x}; \alpha^{(k)}, \boldsymbol{x}_0)$. $\boldsymbol{w} \leftarrow \lambda \boldsymbol{u} + (1 - \lambda)(\boldsymbol{x}_p, t_p)$, where λ solves $f\big(\lambda \boldsymbol{x}_u + (1 - \lambda)\boldsymbol{x}_p\big) - \big(\lambda t_u + (1 - \lambda) t_p\big) = 0$. Compute separating hyperplane, $\ell(\boldsymbol{x}, t) := g(\boldsymbol{w}) + \nabla g(\boldsymbol{w})\big((\boldsymbol{x}, t) - \boldsymbol{w}\big) \leq 0$.

4. *Enumerate Vertices:* $\mathcal{U}^{(k+1)} \leftarrow (\mathcal{U}^{(k)} \cup \mathcal{H}^+) \backslash \mathcal{H}^-$, where $\mathcal{H}^- := \{(\boldsymbol{x}, t) \in \mathcal{U}^{(k)} : \ell(\boldsymbol{x}, t) > 0\}$,

and $\mathcal{H}^+ := \text{extr}\left(\{(\boldsymbol{x}, t) \in \text{conv}(\mathcal{U}^{(k)}) : \ell(\boldsymbol{x}, t) \leq 0\}\right)$, where $\text{extr}(\mathcal{S})$ is the set of extreme points of a set $\mathcal{S}$ and $\text{conv}(\mathcal{S})$ denotes the convex hull of the set $\mathcal{S}$.

5. *Evaluate Underestimation:*
$\alpha^{(k)} \leftarrow \min_{(\boldsymbol{x}_v, t_v) \in \mathcal{H}^+} \left\{ \frac{f(\boldsymbol{x}_v) - \left(f(\boldsymbol{x}_0) + \nabla f(\boldsymbol{x}_0)(\boldsymbol{x}_v - \boldsymbol{x}_0)\right)}{\frac{1}{2}(\boldsymbol{x}_v - \boldsymbol{x}_0)^\top \nabla^2 f(\boldsymbol{x}_0)(\boldsymbol{x}_v - \boldsymbol{x}_0)} \ : \ f(\boldsymbol{x}_v) - q(\boldsymbol{x}_v; \alpha^{(k)}, x_0) < -\varepsilon \right\}$.

6. *Update* $\mathcal{A} : \mathcal{A}^{(k+1)} \leftarrow \{(\boldsymbol{x}, t) \in (\mathcal{A}^{(k)} \cup \mathcal{H}^+) \backslash \mathcal{H}^- : t - q(\boldsymbol{x}; \alpha^{(k)}, \boldsymbol{x}_0) < -\varepsilon\}$.

7. *Iterate:* $k \leftarrow k + 1$. Go to Step 2.

2.1.1 Illustrative Example

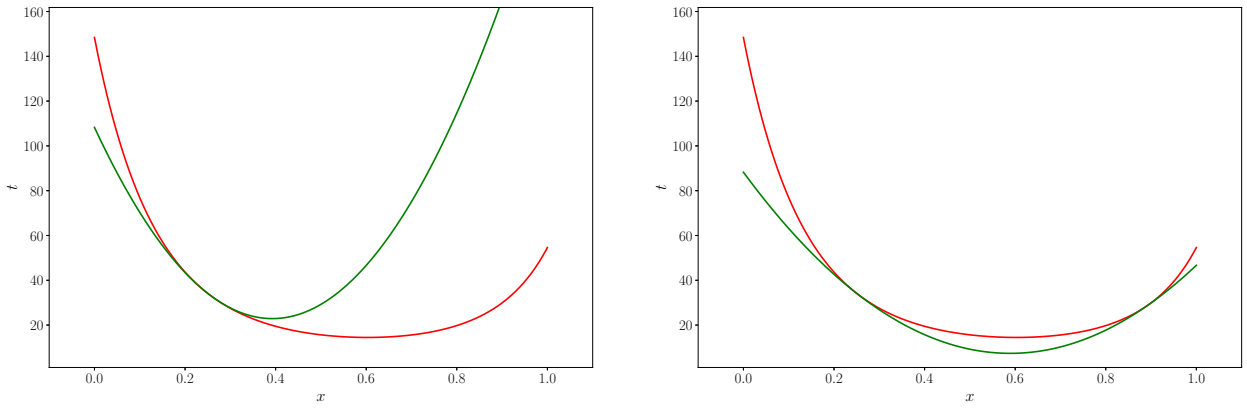


Figure 2. The example function $f(x) = e^{2x^3+4x^2-7x+5}$ in red, and its quadratic underestimator (for $x_0 = 0.25$) in green at initialization (left) and termination (right) of the cutting-plane algorithm.

We demonstrate Algorithm 2.1 using the example function $f(x) = e^{2x^3+4x^2-7x+5}$ from Tawarmalani and Sahinidis [2005] and $x_0 = 0.25$ as the point of construction. Initially, with no modification to the second-order term of the Taylor series approximation ($\alpha = 1$), the left plot in Figure 2 depicts the quadratic's substantial overestimation of $f(x)$. However, once the cutting-plane algorithm terminates after 22 iterations, it was determined that the the second-order term should be scaled by $\alpha = 0.42101$ to yield the final valid quadratic underestimator shown in the plot on the right.

Figure 3 illustrates the steps of the first iteration of Algorithm 2.1. As part of the initialization, the quadratic underestimator is evaluated for overestimation at the vertices $\mathcal{U}^{(0)}$ (solid squares), and

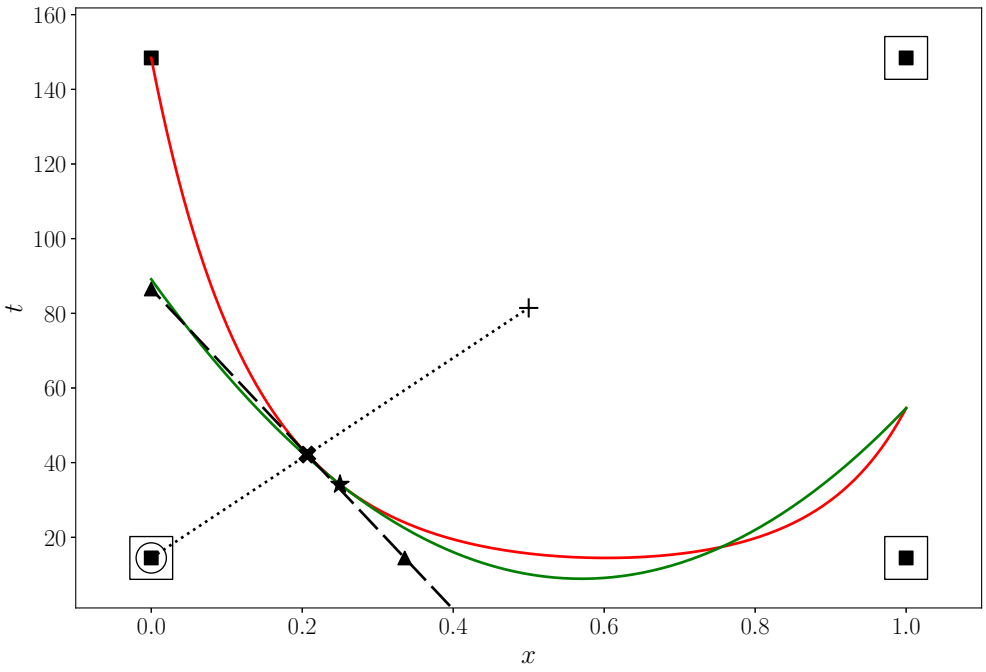


Figure 3. The example function $f(x) = e^{2x^3+4x^2-7x+5}$ (red) and the quadratic underestimator (green) after one iteration of the cutting-plane algorithm. The four solid squares represent $\mathcal{U}^{(0)}$, three of which enclosed in larger squares represent $\mathcal{A}^{(0)}$, the "+" symbol depicts the interior point (x_p, t_p), the star is the point of construction $(x_0, f(x_0))$, the circled solid square is $\boldsymbol{u}$ in Step 3 as well as the sole element of set $\mathcal{H}^-$, the triangles are the set $\mathcal{H}^+$, the dashed line connecting these triangles is $\ell(x,t)$, the dotted line is $\boldsymbol{u} - (\boldsymbol{x}_p, t_p)$, while the "×" where these two lines intersect is $\boldsymbol{w}$ in Step 3.

α is decremented to the value of 0.44641. In Step 2, the convergence criteria is not met as the lower bound of -74.64228 is substantially negative (beyond the tolerance) and the algorithm proceeds to Step 3. There, it identifies the element of $\mathcal{A}^{(0)}$ (squares enclosed by squares) corresponding to the lower bound, $\boldsymbol{u}$ (encircled solid square), the intersection of $\boldsymbol{u}$ with the interior point, $\boldsymbol{w}$ (marked as "×"), and the separating hyperplane, $\ell(x,t)$ (dashed line connecting the triangles). In Step 4, the algorithm computes the intersection of the hyperplane with the convex hull of $\mathcal{U}^{(0)}$ and produces $\mathcal{H}^+$ (triangles), the vertices to be added to $\mathcal{U}^{(0)}$, and $\mathcal{H}^-$ (encircled solid square), the vertices to be removed from $\mathcal{U}^{(0)}$. After generating vertices $\mathcal{H}^+$ to add to $\mathcal{U}^{(0)}$, for the outer approximation, Step 5 evaluates the quadratic underestimator at each newly generated vertex (triangles) and

modifies α, as necessary, to ensure underestimation. In this specific instance, α is not modified because $q(x; \alpha^{(0)}, x_0)$ does underestimate f at both newly generated points (minimizer in Step 5 remains the same). Step 6 computes the set $\mathcal{A}^{(1)}$ by adding the vertices in $\mathcal{H}^+$, removing the vertices from $\mathcal{H}^-$, and only retaining the vertices in $\mathcal{H}^+$ where the objective function evaluates below $-\varepsilon$. Finally, Step 7 prepares the algorithm to execute its next iteration.

2.1.2 Remarks

We make the following remarks concerning the cutting-plane algorithm.

Remark 1. *For a valid underestimator of form (2), the optimal value of (4) is exactly 0 and is attained at $\boldsymbol{x} = \boldsymbol{x}_0$. The main task of the cutting-plane algorithm is to decrease α and increase the lower bound until the quadratic is guaranteed to overestimate the function by at worst ε over the domain.*

Remark 2. *In Step 1, $t_L = \min_{\boldsymbol{x} \in \mathcal{B}} f(\boldsymbol{x})$ is a convex minimization problem, which is robustly and efficiently solvable for small n using readily available convex optimization algorithms, while $t_U = \max_{\boldsymbol{x} \in \mathcal{B}} f(\boldsymbol{x})$ maximizes a convex function over a convex feasible set, which attains its maximum at a vertex. Hence, $t_U = \max_{\boldsymbol{v} \in \mathcal{V}} f(\boldsymbol{v})$.*

Remark 3. *The cutting-plane algorithm requires a point from the interior of the epigraph set, $\{(\boldsymbol{x}_p, t_p) \in \mathcal{B} \times [t^L, t^U] : f(\boldsymbol{x}_p) < t_p\}$. We select the center point of this domain as our interior point inasmuch as it is well-defined, easy to compute, and applicable for all twice-differentiable convex functions. We highlight that, while selecting a different interior point would potentially impact the number of iterations to convergence, it will not impact the final underestimator results.*

Remark 4. *The tolerance ε can be adjusted to reflect the acceptable limit of overestimation and introduces a tradeoff between algorithm efficiency and underestimation correctness. For any function, subtracting the final lower bound from the quadratic provides an underestimator that is guaranteed to underestimate the function over the full domain.*

Remark 5. *In Step 2, we only need to evaluate the objective of (4) for each vertex in $\mathcal{A}^{(k)}$ (rather than $\mathcal{U}^{(k)}$) because Observation 1 implies that any vertex where $t - q(\boldsymbol{x}; \alpha^{(k)}, \boldsymbol{x}_0) >= -\varepsilon$, i.e., the underestimating vertices in $\mathcal{U}^{(k)}$, will never satisfy $t - q(\boldsymbol{x}; \alpha^{(k)}, \boldsymbol{x}_0) < -\varepsilon$ at any future iteration of the algorithm. Hence, using $\mathcal{A}^{(k)}$ reduces the number of computations required for each iteration and thereby expedites the algorithm.*

Remark 6. *Step 3 requires solving the equation* $f(\lambda \boldsymbol{x}_u + (1-\lambda)\boldsymbol{x}_p) - (\lambda t_u + (1-\lambda)t_p) = 0$ *for computing scalar* $\lambda \in (0,1)$*. This can be achieved efficiently using the bisection method.*

Remark 7. *Step 4 enumerates vertices using the procedure outlined in Chen et al. [1991]. We note that this procedure must maintain a set of vertices that are non-degenerate. For a cutting plane generated by the algorithm, defined as* $A\boldsymbol{v} + b \leq 0$*, an existing vertex* $\boldsymbol{v}^*$ *becomes degenerate if* $A\boldsymbol{v}^* + b = 0$*. Whenever such degeneracy occurs, we eliminate it by applying a small perturbation to* b*, similar to the approach described in Chen et al. [1991].*

Remark 8. *In Step 5, we note that the computation* $\alpha^{(k)} = \frac{f(\boldsymbol{x}_v) - \left(f(\boldsymbol{x}_0) + \nabla f(\boldsymbol{x}_0)(\boldsymbol{x}_v - \boldsymbol{x}_0)\right)}{\frac{1}{2}(\boldsymbol{x}_v - \boldsymbol{x}_0)^\top \nabla^2 f(\boldsymbol{x}_0)(\boldsymbol{x}_v - \boldsymbol{x}_0)}$ *is not defined where the denominator is zero, i.e.,* $(\boldsymbol{x}_v - \boldsymbol{x}_0)^\top \nabla^2 f(\boldsymbol{x}_0)(\boldsymbol{x}_v - \boldsymbol{x}_0) = 0$*, but highlight that Step 5 only employs this computation when* $f(\boldsymbol{x}_v) - q(\boldsymbol{x}_v; \alpha^{(k)}, x_0) < -\varepsilon$*. In the case that the denominator is indeed zero, the quadratic underestimator at* $\boldsymbol{x}_v$ *is contained in the first-order Taylor series approximation at* $\boldsymbol{x}_0$*, which is guaranteed to underestimate* f *at* $\boldsymbol{x}_v$ *due to the convexity of* f*. Hence, at any point* $\boldsymbol{x}_v$ *where the denominator is zero, the quadratic underestimates* f *and the above formula for updating* $\alpha^{(k)}$ *is never invoked. In practice, due to the finite precision of computation, the absolute value of the denominator must be checked against some small* $\delta > 0$*.*

Remark 9. *As presented, Algorithm 2.1 uses a tolerance on the lower bound of the objective of (4) as its convergence criterion. Alternatively, one can implement a termination criterion based on a time limit, where the lower bound upon termination could be subtracted from the final quadratic expression, thereby becoming the minimum separation distance between the function and the underestimator. As outer approximation algorithms experience diminishing gains in bound improvement as they proceed, imposing a time limit criterion allows the user to determine a balance of underestimator quality versus computational efficiency according to their preference.*

Remark 10. *The quadratic underestimator is inherently defined through a point of construction,* $\boldsymbol{x}_0$*, to be specified prior to the execution of the algorithm. While the overall quality of the underestimator obviously depends on the selection of* $\boldsymbol{x}_0$*, this quality is ultimately evaluated in the broader context of the underestimator's application. For example, when relaxing a nonlinear term featured in an optimization problem, the particular structure of the latter (i.e., the specific constraints and objective) determines the "optimal" selection of* $\boldsymbol{x}_0$*. For our computational experiments, which*

include a diversity of functional forms, we select points of construction using latin-hypercube sampling, which generates points uniformly distributed across the domain. Although this is a natural and generally applicable approach to generate such points, we highlight that, depending on the practitioner's specific problem of interest, exploring alternative strategies for generating points of construction could be beneficial.

2.2 Algorithm Convergence

For a proof on convergence, we refer the reader to Hoffman [1981], where the convergence of a cutting-plane algorithm for solving a concave minimization problem over a convex set (e.g., (4)) is proved. In particular, the authors of that work prove three lemmas as a foundation for their convergence proof: Lemma 1 in Hoffman [1981] states that the sequence of lower bounds of the objective function $\{\phi(x^k)\}$ is monotonically (but not necessarily strictly) increasing, while Lemmas 2 and 3 in Hoffman [1981] deal with the procedure for constructing separating hyperplanes (Step 3). As we use an identical procedure for separating hyperplanes, the proofs for Lemmas 2 and 3 in Hoffman [1981] apply to our algorithm, with the inconsequential difference that the concave minimization problem (4) has a single nonlinear, convex constraint defining the feasible region (rather than multiple, as in the more general case proved in Hoffman [1981]). We adapt and prove Lemma 1 from Hoffman [1981] for our specific case, where in Algorithm 2.1 we dynamically change the concave objective function by updating the value of $\alpha^{(k)}$ at Step 5.

Lemma 1 (adapted from Hoffman [1981])**.** *The sequence of lower bounds* $\{\gamma(x^k, \alpha^{(k)}, \boldsymbol{x}_0) := \min_{x^k \in \mathcal{A}^{(k)}} t - q(\boldsymbol{x}^k; \alpha^{(k)}, \boldsymbol{x}_0)\}$ *is monotonically increasing.*

Proof. At each consecutive iteration of Algorithm 2.1, the feasible set over which $\gamma(x^k, \alpha^{(k)}, \boldsymbol{x}_0)$ is minimized contracts, i.e., if $\mathcal{S}^{(k)} := \text{conv}(\mathcal{U}^{(k)})$, where $\mathcal{U}^{(k)}$ is the set of extreme points comprising the outer approximation of the convex feasible set $\{(\boldsymbol{x}, t) : f(\boldsymbol{x}) - t \leq 0\}$ at iteration k, then $\mathcal{S}^1 \supseteq \mathcal{S}^2 \supseteq \ldots \supseteq \mathcal{S}^k$, thereby ensuring that $\gamma(\boldsymbol{x}^k, \alpha^{(k)}, \boldsymbol{x}_0)$ monotonically increases in the iterations where $\alpha^{(k)}$ is not updated due to the lack of eligible vertices in Step 5. Furthermore, at any iteration where Step 5 proceeds and $\alpha^{(k)}$ is updated, Observation 1 implies that $q(\boldsymbol{x}; \alpha^{(k)}, \boldsymbol{x}_0)$ will monotonically decrease, and thereby monotonically increase $t - q(\boldsymbol{x}; \alpha^{(k)}, \boldsymbol{x}_0)$. Hence, at all iterations of Algorithm 2.1, $\gamma(\boldsymbol{x}^k, \alpha^{(k)}, \boldsymbol{x}_0)$ monotonically increases. □

Considering Lemma 1 together with Lemmas 2 and 3 from Hoffman [1981], which hold for Algorithm 2.1 without further qualification, the convergence of Algorithm 2.1 now follows directly from Theorem 2 in Hoffman [1981]. Highlighting a distinct property of our algorithm from the one in Hoffman [1981], we reiterate the fact that, for a properly constructed underestimator $q(\boldsymbol{x}; \alpha, \boldsymbol{x}_0)$, the global minimum value of (4) is 0 (attained at $\boldsymbol{x} = \boldsymbol{x}_0$), and so the algorithm will terminate only when all of the vertices comprising the outer approximation of the convex feasible set evaluate the objective of (4) at a value that is larger than the specified tolerance, $-\varepsilon$, which equivalently guarantees the quadratic overestimates the function by at most ε.

2.2.1 Effect of Functional Form on Convergence

We qualitatively characterize and provide some intuition for pathological functions that might cause our cutting-plane algorithm to experience slow convergence. The termination of Algorithm 2.1 occurs in Step 2 and requires that all of the vertices of the outer approximation of f, that is $\{(\boldsymbol{x}, t) \in \mathcal{U}^{(k)}\}$, evaluate $t - q(\boldsymbol{x}; \alpha^{(k)}, \boldsymbol{x}_0) > -\varepsilon$. Consequently, if utilizing the scaled curvature at $\boldsymbol{x}_0$ creates large deviations between f and the underestimator at points in the domain away from $\boldsymbol{x}_0$, then the vertices comprising the outer approximation of f need less refinement in the cutting-plane algorithm for $t - q(\boldsymbol{x}; \alpha^{(k)}, \boldsymbol{x}_0) > -\varepsilon$ to hold, thereby requiring fewer iterations of the cutting-plane algorithm and faster convergence. Additionally, the number of cutting planes required to outer approximate f to a certain degree of precision also contributes to the rate at which the termination criterion is satisfied, as more cutting planes may need to be generated to reduce the difference between the outer approximation and the quadratic to within the required tolerance to satisfactorily prove underestimation. These two considerations, concisely reiterated as (i) the degree to which the function deviates from quadratic and (ii) how efficiently the function can be approximated polyhedrally, determine the number of cutting-plane iterations required for convergence. We also note that, as the dimensionality of the domain of the function increases, nonlinear functions require many more cutting planes for precise polyhedral outer approximations. With this qualitative description, we provide illustrative examples in Figures 4 and 5 that highlight the impact of (i) and (ii) on Algorithm 2.1.

More specifically, Figure 4 plots the 2D functions $f(\boldsymbol{x}) = x_1^2 + x_2^2$, $f(\boldsymbol{x}) = x_1^4 + x_2^4$ and $f(\boldsymbol{x}) = x_1^6 + x_2^6$ over the domain $\boldsymbol{x} \in [-1, 1]^2$, for which using the point of construction $\boldsymbol{x}_0 = (0.5, 0.5)$ leads

Algorithm 2.1 to perform 1387, 45, and 36 iterations, respectively. Therefore, it can be implied that, the greater the dissimilarity from the underestimator form (i.e., quadratic) over the domain, the faster the algorithm's convergence. Figure 5 displays three different *quadratic* functions, each of which is its own convex quadratic underestimator. We highlight that the functions in 5a, 5b, and 5c progressively become "more linear" in the sense that the minimum eigenvalue of the quadratic function in one principal direction approaches 0. Applied on these functions with construction point $\boldsymbol{x}_0 = (0.5, 0.5)$, Algorithm 2.1 requires 1387, 804, and 46 iterations, respectively, evidencing the importance of not only the deviation of the function from the final underestimator but also how precisely the function can be outer approximated using few cutting planes. In practice, quadratic functions can be identified early and omitted from the underestimation procedure. Nevertheless, these principles also apply generally to other functions that have topologies similar to quadratics and/or to functions that are challenging for hyperplanes to efficiently outer-approximate.

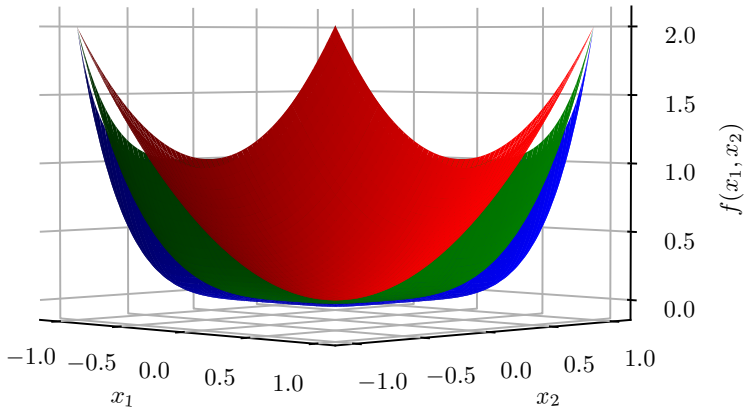
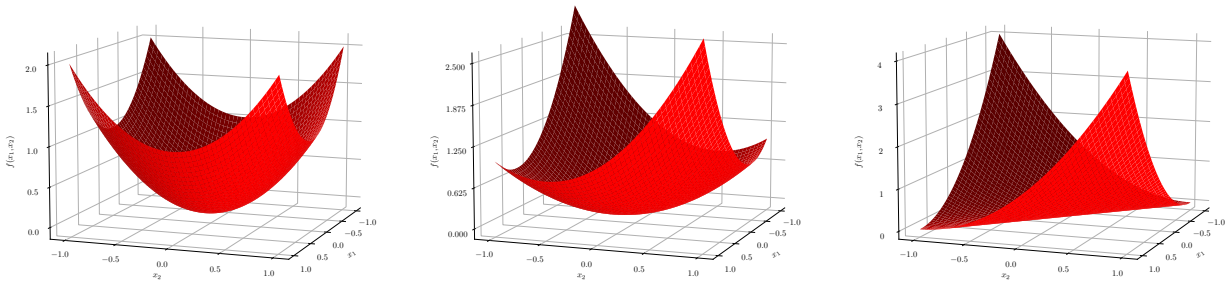


Figure 4. Cross-section of a hierarchy of convex functions that progressively deviate from a perfect quadratic: $f(\boldsymbol{x}) = x_1^2 + x_2^2$ (red), $f(\boldsymbol{x}) = x_1^4 + x_2^4$ (green), $f(\boldsymbol{x}) = x_1^6 + x_2^6$ (blue).

2.3 Properties of Generated Underestimators

After setting forth a preliminary assumption, we establish two important properties of the quadratic underestimators generated by Algorithm 2.1.

Assumption 1. *Given a function $f(\boldsymbol{x}) : \mathbb{R}^n \to \mathbb{R}$ with known bounds on $\boldsymbol{x}$, we assume that, during*



(a) $f(\boldsymbol{x}) = x_1^2 + x_2^2$ **(b)** $f(\boldsymbol{x}) = x_1^2 + x_2^2 + x_1 x_2$ **(c)** $f(\boldsymbol{x}) = x_1^2 + x_2^2 + 2x_1 x_2$

Figure 5. Hierarchy of convex quadratic functions that, from left to right, become progressively more linear in one principal direction.

the execution of Algorithm 2.1, α is decremented (Step 5) at least once.

We note that Assumption 1 generally holds in practice, but we identify two cases where α may not be decremented even once. The first is the case where the tolerance for the algorithm is selected arbitrarily large so that the condition for decrementing α in Step 5 never applies. In this situation, one is advised to select a smaller tolerance and rerun the algorithm. The other case is when $\boldsymbol{x}_0$ is at a point of f where $\nabla^2 f(\boldsymbol{x}) \succcurlyeq \nabla^2 f(\boldsymbol{x}_0)$ for all $\boldsymbol{x} \in \text{dom}(f)$, which implies that $q(\boldsymbol{x}; \alpha, x_0)$ with $\alpha = 1$ underestimates $f(\boldsymbol{x})$ over all points in $\mathcal{B}$.

Property 1. *Under Assumption 1, the quadratic underestimator defined by α will have at least two points that coincide with f.*

Proof. The proof is straightforward. The first point of coincidence is $\boldsymbol{x} = \boldsymbol{x}_0$, by construction. The second point of coincidence is the last point where α was decremented, which is guaranteed to exist by Assumption 1, so that $\exists \boldsymbol{x}' \neq \boldsymbol{x}_0 \in \mathcal{B}$ s.t. $q(\boldsymbol{x}'; \alpha^{(k)}, \boldsymbol{x}_0) = f(\boldsymbol{x}')$ by Step 5 of Algorithm 2.1. Hence, at least two points of coincidence exist.[1] □

Property 2. *The α returned by Algorithm 2.1 is the largest value of α that (i) overestimates $f(\boldsymbol{x})$ at all enumerated vertices by at most ε and (ii) underestimates $f(\boldsymbol{x})$ at all points where α was decremented (i.e., that satisfied the condition of Step 5 during the execution of the algorithm).*

Proof. We first state that, because LB $> -\varepsilon$ at the termination of Algorithm 2.1 and by Step 5 in Algorithm 2.1, the value of α returned satisfies the conditions in Property 2. We prove by

[1] Incidentally, two points of coincidence can be readily identified in Figure 2 (right).

contradiction that there is no larger α. For any $\alpha' > \alpha$, consider the last point at which α was decremented, $\boldsymbol{x}'$, which exists by Assumption 1. Then, because of Lemma 1 and Remark 8, we have that $q(\boldsymbol{x}'; \alpha', \boldsymbol{x}_0) > q(\boldsymbol{x}'; \alpha, \boldsymbol{x}_0)$. In Step 5 at $\boldsymbol{x}'$, we have $q(\boldsymbol{x}'; \alpha, \boldsymbol{x}_0) = f(\boldsymbol{x}')$, and therefore $q(\boldsymbol{x}'; \alpha', \boldsymbol{x}_0) > f(\boldsymbol{x}')$, which contradicts the second condition in Property 2 that $q(\boldsymbol{x}'; \alpha, \boldsymbol{x}_0)$ underestimates $f(\boldsymbol{x})$ at all points where α was decremented during the execution of the algorithm. □

Remark 11. *In the case where $q(\boldsymbol{x}; 1, \boldsymbol{x}_0)$ is a valid underestimator of f and Assumption 1 does not hold, we note that, although Property 1 may not necessarily apply, Property 2 does because the quadratic underestimates the function at all points in the domain ($\nabla^2 f(\boldsymbol{x}) \succcurlyeq \nabla^2 f(\boldsymbol{x}_0)\ \forall \boldsymbol{x} \in \mathbb{R}^n$).*

Remark 12. *In Step 5 of Algorithm 2.1, a practioner may be inclined to update α and allow the quadratic to overestimate the function by the specified convergence tolerance, i.e., $\alpha^{(k)} \leftarrow \frac{f(\boldsymbol{x}_v)+\varepsilon-\left(f(\boldsymbol{x}_0)+\nabla f(\boldsymbol{x}_0)(\boldsymbol{x}_v-\boldsymbol{x}_0)\right)}{\frac{1}{2}(\boldsymbol{x}_v-\boldsymbol{x}_0)^\top \nabla^2 f(\boldsymbol{x}_0)(\boldsymbol{x}_v-\boldsymbol{x}_0)}$, thereby eliminating the second condition in Property 2 requiring underestimation at all points where α was decremented. However, while this approach may seem intuitive to maximize the quadratic underestimator, we remark that it yields a final returned α that would certainly overestimate the last decremented point by ε, thereby invalidating Property 1.*

3 Computational Results

3.1 Computational Experiment Details

For our computational study, we extract all the nonlinear functions from the 233 MINLPLib (`https://www.minlplib.org/`) problems using the filtering criteria `CONVEX==TRUE` and `ProblemType == {MBNLP, MINLP}` applied to the downloadable problem property data sheet. The filtering criteria ensure that, after relaxing the discrete variables, all nonlinear functions in the problems are convex. We instantiate a Pyomo [Hart et al., 2011, 2017] model of the problem instance and execute feasibility-based bounds tightening (fbbt) [Belotti et al., 2010, Puranik and Sahinidis, 2017] using the fbbt implementation available in Pyomo with default parameter values to create realistic bounds for our quadratic underestimation procedure. We further analyze each separable nonlinear expression and decompose it into sub-expressions, each providing an additional nonlinear expression candidate for underestimation. In total, we identify over 9,000 candidate expressions defined

in no larger than 4 dimensions; however, many problems have repeated expressions of similar functional forms, and we therefore reduce the set of expressions into a final set of 19 representative functional forms, defined in 1 to 4 dimensions. In the Appendix, Tables A1–A4 document each expression selected from the MINLPLib, the problem instance it was extracted from, the constraint (or objective) as named in the downloadable `.py` instance file, the precise location of the expression in the constraint (or objective), and the bounds used for the quadratic underestimation procedure as informed by executing fbbt on the problem instance. We note that, among all the candidate expressions, there is at least one occurrence of each functional form with bound values assigned to each variable; therefore, we adopted those and did not have to assign any values for the bounds ourselves. Furthermore, while the functions listed in these tables are mathematically equivalent to the originals found in the MINLPLib problems, they are presented with some algebraic manipulations (e.g., constants and variables factored out of expressions). Here, we emphasize that the performance of our algorithm does not depend on the symbolic representation of the expression and, thus, these manipulations have no impact on our results.

While the functions resulting from the above process provide representative nonlinear functions for quadratic underestimation with different parameterizations that capture nonlinearity within functional forms, they do not include many known convex functional forms that arise in global optimization problems (e.g., sums of even powered polynomials). To that end, we augment our function library with functions extracted from the COCONUT library (`https://arnold-neumaier.at/glopt/coconut/Benchmark/Benchmark.html`), which is a compilation of optimization problems from three different benchmark libraries, namely GlobalLib, CUTE, and constraint satisfaction test problems (CSTP). From those, we specifically identify functional forms that do not appear in the convex MINLPLib test set, and include them in our study. Tables A1–A4 include these functions with the same descriptions as with the MINLPLib-originated ones, with the exception that the location of the expressions are in reference to the GAMS [Bussieck and Meeraus, 2004] file of the instance problem. We remark that many of the problems in the COCONUT library do not have bounds imposed on variables, and thus we assign bounds for variables ensuring that the nonlinearities are captured. All bounds assigned to variables are explicitly noted in the tables. From the COCONUT library, we extract an additional 12 functions, for a grand total of 31 functions for our quadratic underestimation computational study.

We scale the values of each function to lie within the interval $[-1, 1]$ by using the scaling factor $1/\max\{|\min_{\boldsymbol{x}\in\mathcal{B}} f(\boldsymbol{x})|, |\max_{\boldsymbol{x}\in\mathcal{B}} f(\boldsymbol{x})|\}$, and set our convergence tolerance $\varepsilon = 1\text{e}{-3}$. Note that, since the tolerance is applied after such function scaling, it effectively acts as a relative tolerance for all functions.

For each function, we generate 5 points of construction using Latin hypercube sampling and construct quadratic underestimators at each point. We evaluate the quality of our underestimators using the tightness metric defined by Equation (5), which computes the fractional reduction of the hypervolume between a linear underestimator ℓ and the function f that results as a consequence of using a quadratic (as opposed to the linear) underestimator q, constructed at the same point. In order to compute this metric, we discretize the space using $1000n$ Latin hypercube-sampled points, where n is the dimensionality of the function, which we find to be sufficient for our study inasmuch as computational evidence indicated stability in the metric compared to the case of taking into account only $500n$ sample points instead.

$$M_{\ell(\boldsymbol{x})}^{q(\boldsymbol{x})} := \frac{\int_{\boldsymbol{x}\in\text{dom}(f)} \big(q(\boldsymbol{x}) - \ell(\boldsymbol{x})\big) d\boldsymbol{x}}{\int_{\boldsymbol{x}\in\text{dom}(f)} \big(f(\boldsymbol{x}) - \ell(\boldsymbol{x})\big) d\boldsymbol{x}} \tag{5}$$

While we use Equation (5) as a metric for comparing single linear and quadratic underestimators, we note that, when constructing relaxations for optimization problems, this comparison does not consider how the relaxation tightness affects the overall computational time of solving the problem. These considerations would touch upon solver-specific details such as, for example, the tradeoff of adding many rows to the optimization model in the linear relaxation case versus fewer rows in the quadratic relaxation case, and are thus out of the scope of this manuscript.

Moreover, we implemented the algorithm in Python and acknowledge that efficiency gains could be achieved by utilizing a compiled language. Hence, we do not speculate on the impact on computational efficiency of solving optimization problems using quadratic versus linear relaxations. We do note, however, that the rapid development of algorithms solving quadratic problems is continuously reducing the performance gap between linear and quadratic solvers. The computational study is executed on a Lenovo ThinkPad T490 laptop equipped with a 1.80GHz Intel(R) Core(TM) i7-8565U CPU on a Ubuntu 22.04 virtual machine with 8GB RAM and 4 logical processors.

3.2 Results

We report in Table 1 the average and range of the tightness metric, CPU time, and number of vertices enumerated for the nonlinear expressions, categorized by dimension. For 1D and 2D functions, we observe an average metric of 0.534 and 0.570, respectively, while for 3D and 4D functions we observe a somewhat smaller average metric of 0.395 and 0.375, respectively. The diminishing quality of the underestimators in higher dimensions can be explained in part by the rigid modification of the second-order term, where each element of $\nabla^2 f(\boldsymbol{x}_0)$ is identically scaled, thus requiring the choice of α to be sufficiently low to underestimate in all principal directions, and thereby limited by the direction of least curvature. Notwithstanding, we observe on average 35% tighter underestimators for 4-dimensional functions compared to linear underestimators. Table 1 also evidences the efficient construction of the quadratic underestimators. The majority of the underestimators are very efficient to construct, requiring less than 33 CPU milliseconds (ms) for functions in 3D and below. The longest time to construct an underestimator in the 4D case required 182 ms but, on average, functions in 4 dimensions required 51 ms for construction. The number of vertices generated by the algorithm shows an exponential increase in relationship to the dimension and manifests the main tractability limitation.

Table 1. Quadratic underestimator statistics, by function dimension.

Dimension	# Functions	# Underestimators	Avg. Metric	Range Metric	Avg. CPU (ms)	Range CPU (ms)	Avg. # Vertices	Range # Vertices
1	14	70	0.534	0.000–0.899	4	1–11	16.3	4–28
2	8	40	0.570	0.152–0.974	9	4–25	50.5	24–148
3	6	30	0.395	0.059–0.721	11	1–33	95.9	16–316
4	3	15	0.375	0.100–0.781	51	2–182	533.1	32–1951

To gain further insights, we categorize the computational results by ranges of metric, CPU time, and number of enumerated vertices in Table 2. The fact that CPU time is, on average, higher in instances yielding the lowest metric values indicates that additional time expended in the algorithm may not yield tighter underestimators, but also suggests that generating tight underestimators need not require a greater amount of computational expense. Breaking down the results by CPU time evidences the computational efficiency of the algorithm, where approximately 74% of the underestimators required less than 10 ms to generate. Intuitively, the results also manifest the positive correlation of CPU time with the number of enumerated vertices in the cutting-plane

algorithm.

Table 2. Quadratic underestimator statistics by range of metric, CPU time, and number of vertices

Parameter	Range	# Underestimators	Avg. Metric	Avg. CPU (ms)	Avg. # Vertices
Metric	0.00–0.25	36	0.131	23	222.1
	0.25–0.50	44	0.407	7	40.4
	0.50–0.75	40	0.650	10	77.2
	0.75–1.00	35	0.830	7	34.1
CPU Time (ms)	<5	63	0.445	3	18.8
	5–10	52	0.548	7	34.0
	>10	40	0.529	31	277.0
# Vertices	<20	59	0.489	3	14.6
	20–50	54	0.542	6	31.8
	>50	42	0.466	30	272.7

In addition, in order to quantify the variability of the tightness metric across different points of construction, for each function we compute the standard deviation of the metric and normalize it against the mean across all points of construction. Averaging across all functions, we derive a value of 47.4%, which indicates significant variability in the tightness metric for different points of construction. This result alludes to the likely benefit of constructing underestimators at more than one point of construction in practice. Finally, to visually appreciate the tightness of the produced underestimators, we provide some depictions of underestimators for 1D and 2D functions in Figures 6 and 7, respectively.

4 Enhanced Underestimators Within Known Polyhedral Regions

In the context of DGO algorithms that utilize underestimation, the optimization problem often includes a subset of linear constraints, which define a polyhedral region. We can exploit knowledge of such a region to construct even tighter quadratic underestimators, where we allow them to overestimate functions in points that otherwise violate these linear constraints.

More specifically, given a set of such "external" linear constraints $L_j(\boldsymbol{x}) \leq 0$, where j indexes

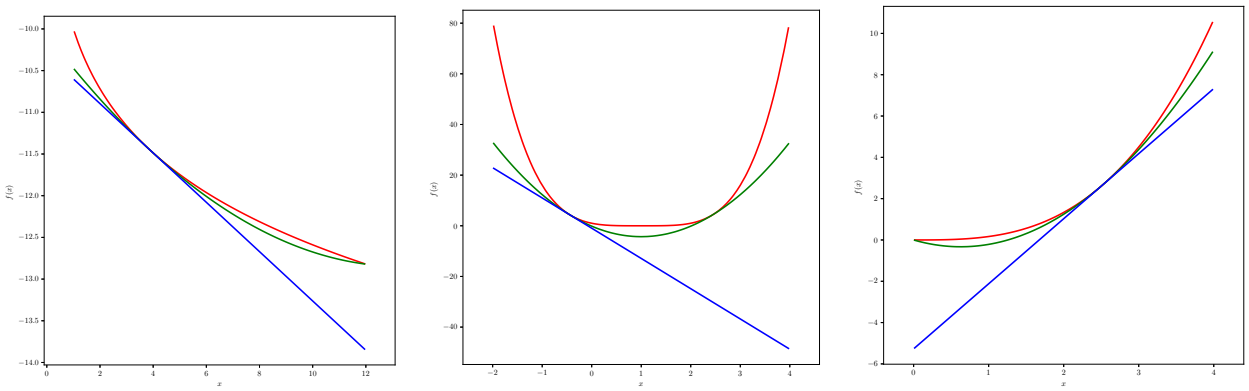


Figure 6. 1D quadratic underestimators (green) and linear underestimators (blue) for functions (red) from chakra (left), hs049 (center), and harker (right) benchmark problems.

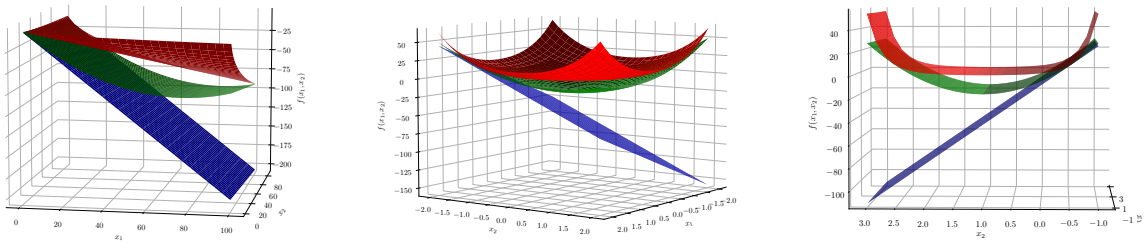


Figure 7. 2D quadratic underestimators (green) and linear underestimators (blue) for functions (red) from tls12 (left), arwhead (center), and polak1 (right) benchmark problems.

over the latter, the following enumerated list captures the relevant requirements and modifications to the cutting-plane algorithm to accommodate this setting:

1. We require that the set of variables of the function that is being underestimated includes all variables referenced in each of the external linear constraints; that is, $\text{dom}(L_j) \subseteq \text{dom}(f)$ for all j.

2. We require that the point of construction selected for the underestimator is within the polyhedral feasible region defined by the linear constraints, which can be enforced through a simple filtering step in preprocessing. This requirement ensures that $\alpha = 1$ is a valid upper bound for the initial value of the scaling parameter in the cutting-plane algorithm.

3. We modify Step 1 of the cutting-plane algorithm by executing the vertex enumeration proce-

dure and intersecting the original box domain defined by bounds on the variables with each external linear constraint *before* lifting the domain in t. We note that redundant constraints are immediately discarded if $\max_{\boldsymbol{x}\in\mathcal{V}} L_j(\boldsymbol{x}) \leq 0$, and that Remark 7 applies in this case as well. After attaining a new vertex set resulting from intersecting the box domain with the linear constraints, we then compute bounds on t. Whereas t^U remains a maximization of a convex function over a convex set, where evaluation of the vertices suffices, in the case of t^L we include the linear constraints in the convex minimization problem, as shown in (6). Here, we remark that including the linear constraints in (6) is not necessary for the algorithm to converge, but can create tighter initial bounds on t that may help the algorithm terminate faster.

$$\begin{aligned} t^L := \quad & \min_{\boldsymbol{x}\in\mathcal{B}} \quad f(\boldsymbol{x}) \\ & \text{s.t.} \quad L_j(\boldsymbol{x}) \leq 0 \quad \forall j \end{aligned} \tag{6}$$

Note how, after the above modification, the intersection of the linear constraints with the original box domain generates a set of vertices that are restricted to the feasible domain. Hence, in the course of the cutting-plane algorithm's remaining execution, as the separating hyperplanes are iteratively intersected with this set of vertices that is already restricted to the feasible domain, the cutting-plane algorithm will never check if the quadratic underestimator overestimates the function (Step 5) at a point that is outside of the feasible domain as informed by the external linear constraints. The overall impact is the generation of tighter underestimators, with the accompanying cost of slower convergence as the underestimators will lie closer to the function and the cutting-plane algorithm will require more refinement to elevate the lower bound to the termination tolerance.

We highlight that the added utility of including external linear constraint information in the construction of quadratic underestimators provides a unique advantage to our method compared to those proposed in the literature. Specifically, we note that this capability allows for the construction of very tight relaxations using relatively few underestimators, as compared to other alternatives that have limited information regarding the optimization problem at hand. We also highlight that many global optimization algorithms generate cuts to facilitate convergence. In our extension, linear constraints that are not explicit in the problem formulation but are derived through cut generation procedures, whether valid by feasibility or optimality, can be leveraged in this extension to construct tighter quadratic underestimators.

4.1 Illustrative Example

We provide a simple example to clearly communicate the impact of integrating external linear constraint information into the cutting-plane algorithm to produce tighter underestimators. We consider the function $f(\boldsymbol{x}) := e^{\frac{1}{2}x_1^2+x_2^2+\frac{1}{4}x_1+\frac{1}{4}x_2+1}$ over box $\boldsymbol{x} \in [0,1]^2$, the point of construction $\boldsymbol{x}_0 = (1,1)$, and two constraints: $x_1 + x_2 \geq 1$ and $x_1 - x_2 \leq 0$. We qualitatively show the change in the underestimator in Figure 8. Quantitatively, the value of α computed without information from external linear constraints is $\alpha = 0.3456$, while this value tightens to $\alpha = 0.4351$ when $x_1+x_2 \geq 1$ is added to the problem, and further to $\alpha = 0.5261$ when both $x_1 + x_2 \geq 1$ and $x_1 - x_2 \leq 0$ are added to the problem. We highlight the clear visual overestimation of the function in the infeasible region presented in Figures 8 and 9. This simple example demonstrates the capability of Algorithm 2.1 to take into account external linear constraint information to generate tighter underestimators in the context of nonlinear optimization problems.

5 Conclusions

In this work, we presented a cutting-plane algorithm to compute the scaling parameter, α, in the second-order Taylor series approximation quadratic underestimator parameter α in the quadratic underestimator proposed by Su et al. [2018] for twice-differentiable convex functions. We established the convergence of the algorithm and demonstrated the quality of the quadratic underestimators it produces as well as its overall computational efficiency. More specifically, through a computational study involving representative nonlinear convex functions extracted from literature benchmark problems, we showed that quadratic underestimators reduce, on average, the hypervolume between the underestimated function and corresponding linear underestimators constructed at same points of the domain by 53.3%, 57.5%, 39.9%, and 35.4%, respectively for dimensions 1 through 4. Construction of quadratic underestimators to a convergence tolerance of $\varepsilon = 1\text{e}{-3}$ was shown to require only a modest amount of computational effort in the order of hundredths or thousandths of a second via a rudimentary implementation. In addition, we proposed an extension of our algorithm to take into account external linear constraint information, which would be available in the context of optimization problems, to further tighten the quadratic underestimators by allowing them to overestimate in the infeasible region, and we demonstrated the impact of this

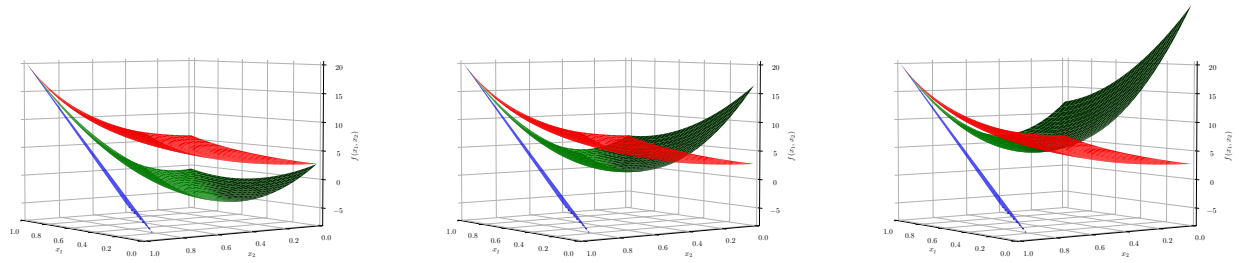


Figure 8. Quadratic underestimators (green) and linear underestimators (blue) for $e^{\frac{1}{2}x_1^2+x_2^2+\frac{1}{4}x_1+\frac{1}{4}x_2+1}$ (red), constructed at point $(1,1)$, in the presence of the following external linear constraints: (i) none (left), (ii) $x_1+x_2 \geq 1$ (center), (iii) $x_1+x_2 \geq 1$ and $x_1-x_2 \leq 0$ (right).

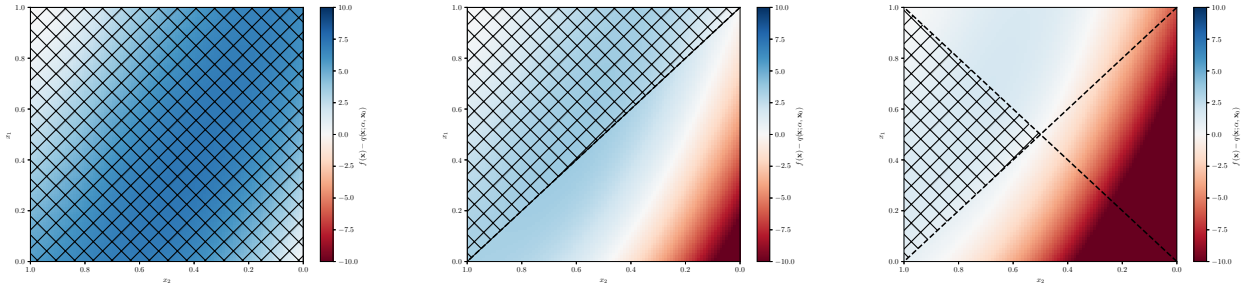


Figure 9. Difference between the function and the underestimator where coincidence (white), underestimation (blue), overestimation (red), and the feasible region (hatched area) of the problem are displayed. With each added constraint (left : none, center : $x_1+x_2 \geq 1$, right : $x_1+x_2 \geq 1$ and $x_1-x_2 \leq 0$), the underestimator becomes tighter as it is allowed to overestimate in the infeasible region.

extension with a small example.

Given the proposed quadratic underestimation procedure to tightly outer approximate twice-differentiable convex functions, there exist opportunities to extend this capability towards outer approximating non-differentiable or non-convex functions, as well. In particular, we highlight the general class of functions representable as a difference-of-convex (d.c.) functions for which our cutting-plane algorithm can be readily adapted. We investigate this methodological avenue in the second part of this two-paper series [Strahl et al.].

Acknowledgments

We acknowledge financial support from Mitsubishi Electric Research Labs (MERL) through the Center for Advanced Process Decision-making (CAPD) at Carnegie Mellon University. William Strahl also gratefully acknowledges support from the R.R. Rothfus Graduate Fellowship in Chemical Engineering and the Chevron Graduate Fellowship in Chemical Engineering. We also acknowledge the contribution of two anonymous referees who helped us improve this manuscript through their insightful comments.

Appendix

Tables A1–A4 herein present the details of the functions used in the computational study of the manuscript.

Table A1. 1D functions in computational study.

Problem	Objective/Constraint	Expression	Bounds on variables	Parameter Values
procurement2mot[a]	e8, 1st term	$f(x_1) = -x_1^a$	x_1^L : 0.10000000000000002	a : 0.333333333333333
			x_1^U : 64.99999999999999	
procurement2mot[a]	e1, 1st term	$f(x_1) = -x_1^a$	x_1^L : 0.10000000000000002	a : 0.5
			x_1^U : 34.99999999999999	
chakra[b]	objcon, 1st term	$f(x_1) = -ax_1^b$	x_1^L : 1	a : 0.774245779798168
			x_1^{U*} : 12	b : 0.75
gtm[b]	objcon, 1st term	$f(x_1) = ax_1^b$	x_1^L : 0.2	a : 8.89583741831423
			x_1^{U*} : 15	b : 0.666666666666667
harker[b]	objcon, 1st cubic term	$f(x_1) = ax_1^b$	x_1^L : 0	a : 0.166666666666667
			x_1^{U*} : 4	b : 3
hs049[c]	objcon, 3rd term	$f(x_1) = -a + x_1^b$	x_1^{L*} : -2	a : 1
			x_1^{U*} : 4	b : 4
schwefel[d]	con1, 1st term	$f(x_1) = x_1^a$	x_1^L : -0.5	a : 10
			x_1^U : 0.4	
srcpm[b]	objcon, 1st term	$f(x_1) = ax_1^{-b}$	x_1^L : 2	a : 1808.40439881057
			x_1^{U*} : 25	b : 2.33333333333333
fo7_ar4_1[a]	e265, 1st term	$f(x_1) = ax_1^{-b}$	x_1^L : 1.5	a : 9
			x_1^U : 6	b : 1
batchs101006m[a]	obj, 11th term	$f(x_1) = a\exp(bx_1)$	x_1^L : 6.514978663740184	a : 150
			x_1^U : 9.615805480084319	b : 0.5
gams01[a]	obj, 4th term	$f(x_1) = a\exp(-bx_1)$	x_1^L : 0	a : 985
			x_1^U : 13.02908889125893	b : 0.306392275145139
polak3[c]	con2, 1st term	$f(x_1) = \exp((-a + x_1)^2)$	x_1^{L*} : -1.25	a : 0.9092974268256817
			x_1^{U*} : 3	
cvxnonsep_nsig20r[a]	e2, 1st term	$f(x_1) = -a\log(x_1)$	x_1^L : 1	a : 0.065
			x_1^U : 10	
odfits[c]	objcon, 1st term	$f(x_1) = ax_1\log(bx_1)$	x_1^L : 0.1	a : 0.5
			x_1^{U*} : 10	b : 0.011111111111111112

[a]MINLPLib, [b]COCONUT/GlobalLib, [c]COCONUT/CUTE, [d]COCONUT/CSTP, [*]bounds assigned

Table A2. 2D functions in computational study.

Problem	Objective/Constraint	Expression	Bounds on variables	Parameter Values
gams01[a]	e103, 1st term	$f(x_1,x_2) = \sqrt{b(-cx_1 - cx_2 + 1)^2 + (dx_1 + dx_2 - 1)^2}$	x_1^L : 10.500000000000032	a : 1255.17137031437
			x_1^U : 18.45973612666075	b : 0.463731258833818
			x_2^L : 0	c : 0.0308163056574352
			x_2^U : 7.959736126660783	d : 0.0421228975436196
tls12[a]	e373, 1st term	$f(x_1,x_2) = -\sqrt{x_1 x_2}$	x_1^L : 1	
			x_1^U : 100	
			x_2^L : 1	
			x_2^U : 86.0	
cvxnonsep_pcon20r[a]	e2, 1st term	$f(x_1,x_2) = a^{(x_1+x_2)}$	x_1^L : 0	a : 2
			x_1^U : 5	
			x_2^L : 0	
			x_2^U : 5	
arwhead[d]	obj, 1st nonlinear term	$f(x_1,x_2) = (x_1^2 + x_2^2)^2$	x_1^{L*} : -2	
			x_1^{U*} : 2	
			x_2^{L*} : -2	
			x_2^{U*} : 2	
batch0812[a]	obj, 8th term	$f(x_1,x_2) = a\exp(x_1 + bx_2)$	x_1^L : 0.15678251173242852	a : 1200
			x_1^U : 1.6094379124341	b : 0.6
			x_2^L : 7.401311768742739	
			x_2^U : 8.00636756765025	
batch0812[a]	e193, 1st term	$f(x_1,x_2) = a\exp(-x_1 + x_2)$	x_1^L : 5.81464756191388	a : 485000
			x_1^U : 5.9395048081772694	
			x_2^L : 0.506817602368452	
			x_2^U : 0.6316748486318415	
polak1[c]	con1, 1st term	$f(x_1,x_2) = \exp(ax_1^2 + (-b + x_2)^2)$	x_1^{L*} : -1	a : 0.001
			x_1^{U*} : 3	b : 1
			x_2^{L*} : -1	
			x_2^{U*} : 3	
synthes2[a]	e1, 1st term	$f(x_1,x_2) = -\log(x_1 + x_2 + 1)$	x_1^L : 0	
			x_1^U : 5.5	
			x_2^L : 0	
			x_2^U : 2.75	

[a]MINLPLib, [b]COCONUT/GlobalLib, [c]COCONUT/CUTE, [d]COCONUT/CSTP, *bounds assigned

Table A3. 3D functions in computational study.

Problem	Objective/Constraint	Expression	Bounds on variables	Parameter Values
p_ball_10b_5p_2d_h[a]	e41, 1st term	$f(x_1,x_2,x_3) = (ax_2+b)\left((\frac{-x_1}{ax_2+b}+c)^2 + d(\frac{-ex_3}{ax_2+b}+1)^2 - 1\right)$	x_1^L : 0	a : 0.9999
			x_1^U : 6.02957539217313	b : 0.0001
			x_2^L : 0	c : 0.648386267690458
			x_2^U : 1	d : 28.5367916413211
			x_3^L : 0	e : 0.187196372132791
			x_3^U : 10	
clay0203hfsg[a]	e106, 1st term	$f(x_1,x_2,x_3) = (ax_2+b)\left(\frac{cx_1^2}{(x_2+d)^2} - \frac{ex_1}{ax_2+b} + \frac{fx_3^2}{(x_2+d)^2} - \frac{gx_3}{ax_2+b}\right)$	x_1^L : 0	a : 0.999
			x_1^U : 18.5	b : 0.001
			x_2^L : 0	c : 1.00200300400501
			x_2^U : 1	d : 0.001001001001001
			x_3^L : 0	e : 35
			x_3^U : 13.0	f : 1.00200300400501
				g : 14
rsyn0805hfsg[a]	e376, 1st term	$f(x_1,x_2,x_3) = (ax_2+b)\left(\frac{x_1}{ax_2+b} - \log(\frac{x_3}{ax_2+b}+1)\right)$	x_1^L : 0	a : 0.999
			x_1^U : 2.39789527279837	b : 0.001
			x_2^L : 0	
			x_2^U : 1	
			x_3^L : 0	
			x_3^U : 10.0	
gams01[a]	e24, 1st term	$f(x_1,x_2,x_3) = a\sqrt{b(cx_1+cx_2-dx_3-1)^2 + (ex_1+ex_2+fx_3-1)^2}$	x_1^L : 2.500000000000116	a : 280.391667345843
			x_1^U : 16.42953724218257	b : 0.0726707025480232
			x_2^L : 0	c : 0.0237333496386274
			x_2^U : 13.929537242182592	d : 0.437956808701174
			x_3^L : 0	e : 0.0240319698226927
			x_3^U : 21.260845713230545	f : 0.175920533216081
batchs101006m[a]	obj, 1st term	$f(x_1,x_2,x_3) = a\exp(bx_1+x_2+x_3)$	x_1^L : 5.7037824746562	a : 250
			x_1^U : 8.1605182474775	b : 0.6
			x_2^L : 0	
			x_2^U : 1.79175946922805	
			x_3^L : 0	
			x_3^U : 1.79175946922805	
risk2bpb[a]	obj, 1st term	$f(x_1,x_2,x_3) = -a(x_1^b + ax_2^b + ax_3^b)$	x_1^L : 0.5	a : 0.333333333333333
			x_1^U : 647499.9311999984	b : 0.329
			x_2^L : 0.5	
			x_2^U : 666358.6828799981	
			x_3^L : 0.5	
			x_3^U : 580813.0108799983	

[a]MINLPLib, [b]COCONUT/GlobalLib, [c]COCONUT/CUTE, [d]COCONUT/CSTP, *bounds assigned

Table A4. 4D functions in computational study.

Problem	Objective/Constraint	Expression	Bounds on variables	Parameter Values
p_ball_10b_5p_3d_h[a]	e62, 1st term	$f(x_1,x_2,x_3,x_4) = (ax_2+b)\left(c(\frac{-dx_1}{ax_2+b}+1)^2 + e(\frac{-fx_3}{ax_2+b}+1)^2 + (\frac{-x_4}{ax_2+b}+g)^2 - 1\right)$	x_1^L : 0	a : 0.9999
			x_1^U : 10	b : 0.0001
			x_2^L : 0	c : 77.8089905885703
			x_2^U : 1	d : 0.113366596747911
			x_3^L : 0	e : 90.5954335749978
			x_3^U : 10	f : 0.10506228598212
			x_4^L : 0	g : 0.894770759747333
			x_4^U : 10	
pspdoc[c]	objcon, entire expression	$f(x_1,x_2,x_3,x_4) = \sqrt{x_1^2+(x_2-x_3)^2+1} + \sqrt{x_2^2+(x_3-x_4)^2+1}$	x_1^{L*} : -10	
			x_1^U : 1	
			x_2^{L*} : -10	
			x_2^{U*} : 10	
			x_3^{L*} : -10	
			x_3^{U*} : 10	
			x_4^{L*} : -10	
			x_4^{U*} : 10	
weapon[d]	con13, 13th term	$f(x_1,x_2,x_3,x_4) = a\exp(-bx_4 - cx_3 - dx_2 - ex_1)$	x_1^L : 0	a : 50
			x_1^{U*} : 8	b : 0.05129329438755058
			x_2^L : 0	c : 0.18632957819149348
			x_2^{U*} : 8	d : 0.05129329438755058
			x_3^L : 0	e : 0.06187540371808753
			x_3^{U*} : 8	
			x_4^L : 0	
			x_4^{U*} : 8	

[a]MINLPLib, [b]COCONUT/GlobalLib, [c]COCONUT/CUTE, [d]COCONUT/CSTP, *bounds assigned

References

Pietro Belotti, Sonia Cafieri, Jon Lee, and Leo Liberti. Feasibility-based bounds tightening via fixed points. In *Combinatorial Optimization and Applications: 4th International Conference, COCOA 2010, Kailua-Kona, HI, USA, December 18-20, 2010, Proceedings, Part I 4*, pages 65–76. Springer, 2010.

Christoph Buchheim and Long Trieu. Quadratic outer approximation for convex integer programming with box constraints. In *Experimental Algorithms: 12th International Symposium, SEA 2013, Rome, Italy, June 5-7, 2013. Proceedings 12*, pages 224–235. Springer, 2013.

Michael R Bussieck and Alex Meeraus. General algebraic modeling system (gams). In *Modeling languages in mathematical optimization*, pages 137–157. Springer, 2004.

Michael Lee Bynum, Anya Castillo, Bernard Knueven, Carl Damon Laird, John Daniel Siirola, and Jean-Paul Watson. Decomposition optimization-based bounds tightening problems via graph partitioning. Technical report, Sandia National Lab.(SNL-NM), Albuquerque, NM (United States), 2021.

Gustavo Henrique Batista Cassiolato, Esdras Penêdo de Carvalho, and Mauro Antonio da Silva Sá Ravagnani. An minlp model for the minimization of installation and operational costs in water distribution networks. *Acta Scientiarum. Technology*, 45:e59993–e59993, 2023.

Pey-Chun Chen, Pierre Hansen, and Brigitte Jaumard. On-line and off-line vertex enumeration by adjacency lists. *Operations Research Letters*, 10(7):403–409, 1991.

Steffen Fahr, Alexander Mitsos, and Dominik Bongartz. Simultaneous deterministic global flowsheet optimization and heat integration: Comparison of formulations. *Computers & Chemical Engineering*, 162:107790, 2022.

Chrysanthos E Gounaris, Eric L First, and Christodoulos A Floudas. Estimation of diffusion anisotropy in microporous crystalline materials and optimization of crystal orientation in membranes. *The Journal of Chemical Physics*, 139(12), 2013.

William E Hart, Jean-Paul Watson, and David L Woodruff. Pyomo: Modeling and solving mathematical programs in python. *Mathematical Programming Computation*, 3:219–260, 2011.

William E Hart, Carl D Laird, Jean-Paul Watson, David L Woodruff, Gabriel A Hackebeil, Bethany L Nicholson, John D Siirola, et al. *Pyomo-optimization modeling in python*, volume 67. Springer, 2017.

Karla Leigh Hoffman. A method for globally minimizing concave functions over convex sets. *mathematical Programming*, 20:22–32, 1981.

Reiner Horst and Panos M Pardalos. *Handbook of global optimization*, volume 2. Springer Science & Business Media, 2013.

JL Klepeis, CA Floudas, D Morikis, CG Tsokos, and JD Lambris. Design of peptide analogues with improved activity using a novel de novo protein design approach. *Industrial & engineering chemistry research*, 43(14):3817–3826, 2004.

Ruth Misener, María Fuentes Garí, Maria Rende, Eirini Velliou, Nicki Panoskaltsis, Efstratios N Pistikopoulos, and Athanasios Mantalaris. Global superstructure optimisation of red blood cell production in a parallelised hollow fibre bioreactor. *Computers & Chemical Engineering*, 71: 532–553, 2014.

Hans Mittelmann. Benchmarks for optimization software, 2023. URL `https://plato.asu.edu/bench.html`. Accessed on December 23, 2023.

Sylvain Mouret, Ignacio E Grossmann, and Pierre Pestiaux. A new lagrangian decomposition approach applied to the integration of refinery planning and crude-oil scheduling. *Computers & Chemical Engineering*, 35(12):2750–2766, 2011.

Alireza Olama, Eduardo Camponogara, and Paulo RC Mendes. Distributed primal outer approximation algorithm for sparse convex programming with separable structures. *Journal of Global Optimization*, 86(3):637–670, 2023.

Frances E Pereira, George Jackson, Amparo Galindo, and Claire S Adjiman. A duality-based optimisation approach for the reliable solution of (p, t) phase equilibrium in volume-composition space. *Fluid Phase Equilibria*, 299(1):1–23, 2010.

Yash Puranik and Nikolaos V Sahinidis. Domain reduction techniques for global nlp and minlp optimization. *Constraints*, 22(3):338–376, 2017.

Luis Miguel Rios and Nikolaos V Sahinidis. Portfolio optimization for wealth-dependent risk preferences. *Annals of Operations Research*, 177(1):63–90, 2010.

Susanne Sass, Angelos Tsoukalas, Ian H Bell, Dominik Bongartz, Jaromił Najman, and Alexander Mitsos. Towards global parameter estimation exploiting reduced data sets. *Optimization Methods and Software*, pages 1–13, 2023.

William R Strahl, Arvind Raghunathan, Nikolaos V Sahinidis, and Chrysanthos E Gounaris. Constructing tight quadratic relaxations for global optimization: II. Underestimating difference-of-convex (d.c.) functions. *Forthcoming*.

Matthew Streeter and Joshua V Dillon. Automatically bounding the Taylor remainder series: Tighter bounds and new applications. *arXiv preprint arXiv:2212.11429*, 2022.

Matthew Streeter and Joshua V Dillon. Sharp Taylor polynomial enclosures in one dimension. *arXiv preprint arXiv:2308.00679*, 2023.

Lijie Su, Lixin Tang, David E Bernal, and Ignacio E Grossmann. Improved quadratic cuts for convex mixed-integer nonlinear programs. *Computers & Chemical Engineering*, 109:77–95, 2018.

Mohit Tawarmalani and Nikolaos V Sahinidis. A polyhedral branch-and-cut approach to global optimization. *Mathematical programming*, 103(2):225–249, 2005.