SIAM J. Optim.
Vol. 20, No. 2, pp. 573–601

MCCORMICK-BASED RELAXATIONS OF ALGORITHMS[*]

ALEXANDER MITSOS[†], BENOÎT CHACHUAT[‡], AND PAUL I. BARTON[§]

Abstract. Theory and implementation for the global optimization of a wide class of algorithms is presented via convex/affine relaxations. The basis for the proposed relaxations is the systematic construction of subgradients for the convex relaxations of factorable functions by McCormick [*Math. Prog.*, 10 (1976), pp. 147–175]. Similar to the convex relaxation, the subgradient propagation relies on the recursive application of a few rules, namely, the calculation of subgradients for addition, multiplication, and composition operations. Subgradients at interior points can be calculated for any factorable function for which a McCormick relaxation exists, provided that subgradients are known for the relaxations of the univariate intrinsic functions. For boundary points, additional assumptions are necessary. An automated implementation based on operator overloading is presented, and the calculation of bounds based on affine relaxation is demonstrated for illustrative examples. Two numerical examples for the global optimization of algorithms are presented. In both examples a parameter estimation problem with embedded differential equations is considered. The solution of the differential equations is approximated by algorithms with a fixed number of iterations.

Key words. nonconvex optimization, global optimization, convex relaxation, subgradient, nonsmooth

AMS subject classifications. 65K05, 90C26, 90C30, 49M20

DOI. 10.1137/080717341

1. Introduction.

The development of deterministic algorithms based on continuous and/or discrete branch-and-bound [10, 17, 18] has facilitated the global optimization of nonconvex programs. The basic principle of branch-and-bound, and related algorithms such as branch-and-cut [19] and branch-and-reduce [27], is to bound the optimal objective value between a lower bound and an upper bound. By branching on the host set, these bounds become tighter and eventually converge. For minimization, upper bounds are typically obtained via a feasible point or via a local solution of the original program. For the lower bound, typically a convex or affine relaxation of the nonconvex program is constructed and solved to global optimality via a convex solver. Convex and concave envelopes or tight relaxations are known for a variety of simple nonlinear terms [1, 33, 35], and this allows the construction of convex and concave relaxations for a quite general class of functions through several methods [21, 2, 33, 12]. Simple lower bounds from interval analysis are also widely used in global optimization, e.g., [6, 7, 25]. Such bounds are often weaker but less computationally expensive to evaluate than relaxation-based bounds. For instance, for a box-constrained problem, no linear program (LP) or convex nonlinear program (NLP) needs to be solved.

The majority of the literature on global optimization considers nonconvex programs for which explicit functions are known for the objective and constraints. A more

[*]Received by the editors March 1, 2008; accepted for publication (in revised form) December 16, 2008; published electronically May 22, 2009. This paper is based upon work supported by the National Science Foundation under grant CTS-0521962.
http://www.siam.org/journals/siopt/20-2/71734.html

[†]Aachen Institute for Advanced Study in Computational Engineering Science, RWTH Aachen, Germany (amitsos@alum.mit.edu). Financial support from the Deutsche Forschungsgemeinschaft (German Research Association) through grant GSC 111 is gratefully acknowledged.

[‡]Department of Chemical Engineering, McMaster University, Hamilton, ON (bchachua@alum.mit.edu).

[§]Corresponding author. Department of Chemical Engineering, Massachusetts Institute of Technology, Cambridge, MA 02139 (pib@mit.edu).

general case is that the objective function and/or constraints are calculated implicitly by an algorithm, e.g., solution of a boundary value problem via the finite element method. This article sets the foundations for the global optimization of such problems by automatically constructing convex and/or affine relaxations of algorithms. This concept is similar in principle to that of automatic differentiation (AD) [13] and extends the applicability of global optimization algorithms considerably. If a function is defined implicitly by an algorithm, the propagation of McCormick's composition result [21, 22] results in a (nonsmooth) convex relaxation of the function. In this article the systematic propagation of subgradients is described without introducing auxiliary variables. The calculation of subgradients allows the use of a nonsmooth NLP solver to obtain the required lower bound. Alternatively, a linearization via the subgradients can give an affine relaxation. Throughout this article the latter case is applied to algorithms with a fixed number of iterations. The optimization problems considered are box-constrained, and therefore, the affine relaxation requires a single function and subgradient evaluation.

In addition to the optimization of algorithms, the proposed subgradient propagation can be used in other areas of global optimization. Efficient solvers for convex NLPs involving nonsmooth functions, such as bundle methods, have recently been developed, e.g., [20, 15], and the development given here would allow their application to the lower bounding problem of global optimization when the McCormick relaxations are employed. Due to the scalability and reliability of linear programming solvers, linearization of a smooth reformulation via the introduction of auxiliary variables and constraints has been successfully employed in nonlinear programs and mixed-integer nonlinear programs (MINLPs) [35, 36, 34]. The results of this paper could also be used for NLPs and MINLPs providing very cheap lower bounds, but the introduction of auxiliary variables and constraints might furnish tighter relaxations [34, p. 128]. Therefore, the results proposed herein should be viewed as an alternative to and not as a replacement for existing methods; as is demonstrated below, our proposal is better than existing methods for some problems. However, no claim for superiority in general is made.

It should be noted that the introduction of auxiliary variables is not always advisable or possible. For instance, two recent proposals for the construction of upper bounds for semi-infinite programs (SIPs) rely on the construction of a convex relaxation of the lower-level problem [11, 23]. One of the possibilities proposed in [23] is to further relax the lower-level program by linearizations at arbitrary parameter points. This is only advisable when the set of parameter vertices is easily calculated [23], which, in general, can only be done exactly when no auxiliary variables are introduced. Moreover, the introduction of auxiliary variables makes the upper bounding problem significantly more computationally expensive to solve. Another area where using auxiliary variables seems intractable is the method by Singer and Barton [30] to construct convex relaxations for the solutions of ordinary differential equations (ODEs) dependent on parameters. Their method relies on linearizations to convex and concave relaxations of the right-hand sides of the ODEs via McCormick's composition theorem and factorable presentation without the introduction of additional variables and constraints. In general, this gives a nonsmooth relaxation, and therefore, the linearization requires the computation of subgradients.

Note at this point that, despite a superficial similarity, the subgradients proposed in this article are very different from the construction of piecewise affine relaxations by Wang and Chang [39, 40, 41]. In contrast to the results herein, Wang and Chang do not consider convex relaxation prior to the linearization and do not use subgradient information. For the intended use in SIPs, piecewise linearization is not suitable because the set of parameter vertices cannot be easily calculated. Also, the method proposed by Wang and Chang does not seem applicable to the relaxation of algorithms.

In the remainder of this paper, first the theoretical framework of McCormick relaxations as well as basic results on subgradients are summarized; these results are from the literature. In section 3, additional theoretical results required for the propagation of subgradients of the McCormick relaxations are developed, along with a description of important points for application and illustrative examples; these results are not available in the literature and are required for the relaxations of algorithms. The results are given for a single subgradient as opposed to the entire subdifferential to maintain consistency with the implementation. Presumably, all theoretical results could be extended to the entire subdifferential, but there are severe complications for implementation. In section 4, an automated implementation is described along with illustrative examples. In section 5, the relaxation of algorithms is described in more detail, and two simple numerical examples illustrate its potential. The paper is concluded by discussing potential future work. To our best knowledge the material presented in sections 3, 4, and 5 is original with the exception of subsection 4.1, which gives a brief overview of AD.

2. Background.

This section contains definitions of concepts used in what follows, as well as a summary of the existing literature results needed for the development of the main results.

2.1. Definitions.

Throughout the paper convex and concave relaxations are considered.

Definition 2.1 (relaxation of functions). *Given a convex set $Z \subset \mathbb{R}^n$ and a function $f : Z \to \mathbb{R}$, a convex function $f^u : Z \to \mathbb{R}$ is a* convex relaxation *(or* convex underestimator*) of f on Z if $f^u(\mathbf{z}) \leq f(\mathbf{z})$ for all $\mathbf{z} \in Z$. Similarly, a concave function $f^o : Z \to \mathbb{R}$ is a* concave relaxation *(or* concave overestimator*) of f on Z if $f^o(\mathbf{z}) \geq f(\mathbf{z})$ for all $\mathbf{z} \in Z$. The definitions for* affine underestimator *f^{ul} and* affine overestimator *f^{ol} are analogous.*

McCormick relaxations are applicable to factorable functions.

Definition 2.2 (factorable function). *A function is* factorable *if it is defined by a finite recursive composition of binary sums, binary products, and a given library of univariate intrinsic functions.*

Factorable functions cover a quite general class of functions. The simplest subclass of factorable functions are those defined without recursion, e.g., $g : Z \to \mathbb{R}$, $g(\mathbf{z}) = F_1(f_1(\mathbf{z})) + F_2(f_2(\mathbf{z}))F_3(f_3(\mathbf{z}))$, where $Z \subset \mathbb{R}^n$, $F_1, F_2, F_3 : \mathbb{R} \to \mathbb{R}$ are from the library of univariate intrinsic functions, and $f_1, f_2, f_3 : Z \to \mathbb{R}$. For the application of McCormick's composition theorem, all functions $(F_1, F_2, F_3, f_1, f_2, f_3)$ must have known convex and concave relaxations as well as known enclosures for their ranges.

Definition 2.3 (McCormick relaxations). *The relaxations of a factorable function that are formed via the recursive application of rules for the relaxation of univariate composition, binary multiplication, and binary addition from convex and concave relaxations of the univariate intrinsic functions, without the introduction of auxiliary variables, are termed* McCormick relaxations.

The aforementioned rules are described in detail in section 2.3.

McCormick relaxations are, in general, nonsmooth and, therefore, subgradients are needed.

Definition 2.4 (subgradient). *Let $Z \subset \mathbb{R}^n$ be a nonempty convex set, $f^u : Z \to \mathbb{R}$ be convex, and $f^o : Z \to \mathbb{R}$ be concave. A vector $\mathbf{s}^u \in \mathbb{R}^n$ is called a* subgradient *of*

f^u at $\bar{\mathbf{z}} \in Z$ if $f^u(\mathbf{z}) \geq f^u(\bar{\mathbf{z}}) + (\mathbf{s}^u)^{\mathrm{T}}(\mathbf{z} - \bar{\mathbf{z}})$ for all $\mathbf{z} \in Z$. A vector $\mathbf{s}^o \in \mathbb{R}^n$ is called a subgradient *of f^o at $\bar{\mathbf{z}} \in Z$ if $f^o(\mathbf{z}) \leq f^o(\bar{\mathbf{z}}) + (\mathbf{s}^o)^{\mathrm{T}}(\mathbf{z} - \bar{\mathbf{z}})$ for all $\mathbf{z} \in Z$.*

Existence of subgradients on the interior of Z is guaranteed, and for differentiable convex and concave functions, the unique subgradient is the gradient [4].

In addition to the well-known min and max functions, McCormick relaxations use the mid function, which selects the middle value between three scalar arguments.

DEFINITION 2.5 (mid function). *Given three numbers $\alpha, \beta, \gamma \in \mathbb{R}$, the mid function is defined as*

$$\operatorname{mid}\{\alpha, \beta, \gamma\} = \begin{cases} \alpha & \text{if } \beta \leq \alpha \leq \gamma \text{ or } \gamma \leq \alpha \leq \beta, \\ \beta & \text{if } \alpha \leq \beta \leq \gamma \text{ or } \gamma \leq \beta \leq \alpha, \\ \gamma & \text{if } \alpha \leq \gamma \leq \beta \text{ or } \beta \leq \gamma \leq \alpha. \end{cases}$$

2.2. Interval extensions. A necessary step in constructing convex relaxations via the McCormick factorization is the recursive propagation of estimates of the ranges of the factors. This is typically done via natural interval extensions, which are briefly described here. For a thorough description of interval analysis, the reader is referred to the literature, e.g., [24, 3]. The first step in computing the natural interval extension is to decompose a given function into a finite recursive composition of elementary operations (e.g., binary multiplication, binary addition) and intrinsic functions, such as monomials or the exponential function; compare also the definition of a factorable function. For each of the intrinsic functions and elementary operations, a rule is available to construct the natural interval extension. For instance, in the addition of two intervals, the lower bound of the sum is given by summing the two lower bounds and the upper bound of the sum by summing the two upper bounds. In general, natural interval extensions lead to an overestimation of the range, but in special cases, such as monomials, an exact estimate is obtained.

2.3. McCormick relaxations. Constructing convex and concave relaxations of factorable functions requires rules for the relaxation of sums, products, and univariate composition. The sum of convex functions is convex, and the sum of concave functions is concave. Therefore, convex and concave relaxations for the sum of two functions can be easily constructed as stated in the following proposition.

PROPOSITION 2.6 (relaxations of sums). *Let $Z \subset \mathbb{R}^n$ be a nonempty convex set, and $g, g_1, g_2 : Z \to \mathbb{R}$ such that $g(\mathbf{z}) = g_1(\mathbf{z}) + g_2(\mathbf{z})$. Let $g_1^u, g_1^o : Z \to \mathbb{R}$ be a convex and concave relaxation of g_1 on Z, respectively. Similarly, let $g_2^u, g_2^o : Z \to \mathbb{R}$ be a convex and concave relaxation of g_2 on Z, respectively. Then, $g^u, g^o : Z \to \mathbb{R}$, such that*

$$g^u(\mathbf{z}) = g_1^u(\mathbf{z}) + g_2^u(\mathbf{z}), \qquad g^o(\mathbf{z}) = g_1^o(\mathbf{z}) + g_2^o(\mathbf{z}),$$

are, respectively, a convex and concave relaxation of g on Z.

Calculating relaxations for the product of two functions is somewhat more elaborate, as shown in the following proposition, which follows from the convex and concave envelopes of a bilinear function [21].

PROPOSITION 2.7 (relaxations of products). *Let $Z \subset \mathbb{R}^n$ be a nonempty convex set, and $g, g_1, g_2 : Z \to \mathbb{R}$ such that $g(\mathbf{z}) = g_1(\mathbf{z})\, g_2(\mathbf{z})$. Let $g_1^u, g_1^o : Z \to \mathbb{R}$ be a convex and concave relaxation of g_1 on Z, respectively. Similarly, let $g_2^u, g_2^o : Z \to \mathbb{R}$ be a convex and concave relaxation of g_2 on Z, respectively. Furthermore, let $g_1^L, g_1^U, g_2^L, g_2^U \in \mathbb{R}$ such that*

$$g_1^L \leq g_1(\mathbf{z}) \leq g_1^U \quad \text{for all } \mathbf{z} \in Z \quad \text{and} \quad g_2^L \leq g_2(\mathbf{z}) \leq g_2^U \quad \text{for all } \mathbf{z} \in Z.$$

Consider the following intermediate functions, $\alpha_1, \alpha_2, \beta_1, \beta_2, \gamma_1, \gamma_2, \delta_1, \delta_2 : Z \to \mathbb{R}$*:*

$$\begin{aligned}
\alpha_1(\mathbf{z}) &= \min\left\{g_2^L g_1^u(\mathbf{z}), g_2^L g_1^o(\mathbf{z})\right\}, & \alpha_2(\mathbf{z}) &= \min\left\{g_1^L g_2^u(\mathbf{z}), g_1^L g_2^o(\mathbf{z})\right\},\\
\beta_1(\mathbf{z}) &= \min\left\{g_2^U g_1^u(\mathbf{z}), g_2^U g_1^o(\mathbf{z})\right\}, & \beta_2(\mathbf{z}) &= \min\left\{g_1^U g_2^u(\mathbf{z}), g_1^U g_2^o(\mathbf{z})\right\},\\
\gamma_1(\mathbf{z}) &= \max\left\{g_2^L g_1^u(\mathbf{z}), g_2^L g_1^o(\mathbf{z})\right\}, & \gamma_2(\mathbf{z}) &= \max\left\{g_1^U g_2^u(\mathbf{z}), g_1^U g_2^o(\mathbf{z})\right\},\\
\delta_1(\mathbf{z}) &= \max\left\{g_2^U g_1^u(\mathbf{z}), g_2^U g_1^o(\mathbf{z})\right\}, & \delta_2(\mathbf{z}) &= \max\left\{g_1^L g_2^u(\mathbf{z}), g_1^L g_2^o(\mathbf{z})\right\}.
\end{aligned}$$

Then, α_1*,* α_2*,* β_1*, and* β_2 *are convex on* Z*, while* γ_1*,* γ_2*,* δ_1*, and* δ_2 *are concave on* Z*. Moreover,* $g^u, g^o : Z \to \mathbb{R}$*, such that*

$$\begin{aligned}
g^u(\mathbf{z}) &= \max\left\{\alpha_1(\mathbf{z}) + \alpha_2(\mathbf{z}) - g_1^L g_2^L, \beta_1(\mathbf{z}) + \beta_2(\mathbf{z}) - g_1^U g_2^U\right\},\\
g^o(\mathbf{z}) &= \min\left\{\gamma_1(\mathbf{z}) + \gamma_2(\mathbf{z}) - g_1^U g_2^L, \delta_1(\mathbf{z}) + \delta_2(\mathbf{z}) - g_1^L g_2^U\right\},
\end{aligned}$$

are, respectively, a convex and concave relaxation of g *on* Z*.*

McCormick [21, 22] provided a relaxation result for the composition of functions. Note that, in deviation from these references, any convex/concave relaxation is allowed here for the univariate function without restriction to the convex/concave envelope.

THEOREM 2.8 (McCormick's composition theorem). *Let* $Z \subset \mathbb{R}^n$ *and* $X \subset \mathbb{R}$ *be nonempty convex sets. Consider the composite function* $g = F \circ f$*, where* $f : Z \to \mathbb{R}$ *is continuous,* $F : X \to \mathbb{R}$*, and let* $f(Z) \subset X$*. Suppose that a convex relaxation* $f^u : Z \to \mathbb{R}$ *and a concave relaxation* $f^o : Z \to \mathbb{R}$ *of* f *on* Z *are known. Let* $F^u : X \to \mathbb{R}$ *be a convex relaxation of* F *on* X*, let* $F^o : X \to \mathbb{R}$ *be a concave relaxation of* F *on* X*, let* $x^{\min} \in X$ *be a point at which* F^u *attains its infimum on* X*, and let* $x^{\max} \in X$ *be a point at which* F^o *attains its supremum on* X*. Then,* $g^u : Z \to \mathbb{R}$*,*

$$g^u(\mathbf{z}) = F^u\left(\operatorname{mid}\left\{f^u(\mathbf{z}), f^o(\mathbf{z}), x^{\min}\right\}\right)$$

is a convex relaxation of g *on* Z*, and* $g^o : Z \to \mathbb{R}$*,*

$$g^o(\mathbf{z}) = F^o\left(\operatorname{mid}\left\{f^u(\mathbf{z}), f^o(\mathbf{z}), x^{\max}\right\}\right)$$

is a concave relaxation of g *on* Z*.*

By definition $f^u(\mathbf{z}) \le f^o(\mathbf{z})$, and therefore,

$$\operatorname{mid}\left\{f^u(\mathbf{z}), f^o(\mathbf{z}), x^{\min}\right\} = \begin{cases} f^u(\mathbf{z}) & \text{if } x^{\min} < f^u(\mathbf{z}),\\ f^o(\mathbf{z}) & \text{if } x^{\min} > f^o(\mathbf{z}),\\ x^{\min} & \text{otherwise,}\end{cases}$$

$$\operatorname{mid}\left\{f^u(\mathbf{z}), f^o(\mathbf{z}), x^{\max}\right\} = \begin{cases} f^u(\mathbf{z}) & \text{if } x^{\max} < f^u(\mathbf{z}),\\ f^o(\mathbf{z}) & \text{if } x^{\max} > f^o(\mathbf{z}),\\ x^{\max} & \text{otherwise.}\end{cases}$$

Due to the presence of the mid function, the convex and concave relaxations constructed by application of Theorem 2.8 are not guaranteed to be smooth. This situation arises, e.g., when $x^{\min} \in \operatorname{int}(f^u(Z))$ and is illustrated by the following example.

Example 2.1 (nonsmooth McCormick relaxation). Let

$$Z = \{z \in \mathbb{R} : -1 \le z \le 1\}$$

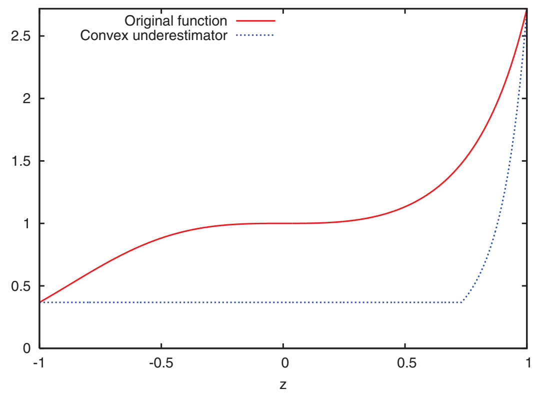


FIG. 2.1. *Graphical illustration of Example* 2.1.

and note that Z is a nonempty and convex set. Consider the function $g : Z \to \mathbb{R}$, $g(z) = e^{z^3}$, which can be written as the composition $g = F \circ f$ by choosing $f(z) = z^3$, $X = [-1, 1]$, and $F(x) = e^x$. A valid underestimator f^u and overestimator f^o of f on Z are given by the αBB relaxations [2]:

$$f^u(z) = z^3 + 3\left(z^2 - 1\right), \qquad f^o(z) = z^3 - 3\left(z^2 - 1\right).$$

Furthermore, since the exponential function is convex, $F^u(x) = F(x) = e^x$. Finally, since the exponential function is increasing, $x^{\min} = -1$. By application of Theorem 2.8, a convex underestimator $g^u : Z \to \mathbb{R}$ of g on Z is given by

$$g^u(z) = e^{\operatorname{mid}\left\{z^3+3\left(z^2-1\right),z^3-3\left(z^2-1\right),-1\right\}} = \begin{cases} e^{-1} & \text{if } z \leq -1+\sqrt{3}, \\ e^{z^3+3\left(z^2-1\right)} & \text{otherwise,} \end{cases}$$

which is clearly nonsmooth, as seen also in Figure 2.1.

2.4. Subgradients. Some results are now summarized for subgradients which can be found in the literature, e.g., [26, 14, 15, 16] or are quite intuitive and can be derived easily. The following proposition is based on Theorem D-4.1.1 from Hiriart-Urruty and Lemaréchal [16].

PROPOSITION 2.9 (addition rule for subgradients). *Let $Z \subset \mathbb{R}^n$ be a nonempty convex set. Suppose that $f_1^u, f_2^u : Z \to \mathbb{R}$ are convex, $\mathbf{s}_1^u$ is a subgradient of f_1^u at $\bar{\mathbf{z}} \in Z$, and $\mathbf{s}_2^u$ is a subgradient of f_2^u at $\bar{\mathbf{z}}$. Then, the function $f_1^u + f_2^u$ is convex on Z, and $\mathbf{s}_1^u + \mathbf{s}_2^u$ is a subgradient of $f_1^u + f_2^u$ at $\bar{\mathbf{z}}$. Similarly, suppose that $f_1^o, f_2^o : Z \to \mathbb{R}$ are concave, $\mathbf{s}_1^o$ is a subgradient of f_1^o at $\bar{\mathbf{z}} \in Z$, and $\mathbf{s}_2^o$ is a subgradient of f_2^o at $\bar{\mathbf{z}}$. Then, the function $f_1^o + f_2^o$ is concave on Z, and $\mathbf{s}_1^o + \mathbf{s}_2^o$ is a subgradient of $f_1^o + f_2^o$ at $\bar{\mathbf{z}}$.*

For the following proposition, see also Theorem D-4.4.1 from Hiriart-Urruty and Lemaréchal [16].

PROPOSITION 2.10 (max and min rule for subgradients). *Let $Z \subset \mathbb{R}^n$ be a nonempty convex set. Suppose that $\alpha_1, \alpha_2, \beta_1, \beta_2 : Z \to \mathbb{R}$ are convex, $\mathbf{s}^{\alpha_1}$ is a subgradient of α_1 at $\bar{\mathbf{z}} \in Z$, $\mathbf{s}^{\alpha_2}$ is a subgradient of α_2 at $\bar{\mathbf{z}}$, $\mathbf{s}^{\beta_1}$ is a subgradient of β_1 at $\bar{\mathbf{z}}$, $\mathbf{s}^{\beta_2}$ is a subgradient of β_2 at $\bar{\mathbf{z}}$, and $a, b \in \mathbb{R}$ are constants. Then, the function $g^{\max} : Z \to \mathbb{R}$*

$$g^{\max}(\mathbf{z}) \equiv \max\{\alpha_1(\mathbf{z}) + \alpha_2(\mathbf{z}) - a, \beta_1(\mathbf{z}) + \beta_2(\mathbf{z}) - b\}$$

is convex on Z and

1. *if $\alpha_1(\bar{\mathbf{z}}) + \alpha_2(\bar{\mathbf{z}}) - a \geq \beta_1(\bar{\mathbf{z}}) + \beta_2(\bar{\mathbf{z}}) - b$, then a subgradient of $g^{\max}$ at $\bar{\mathbf{z}}$ is given by $\mathbf{s}^{\alpha_1} + \mathbf{s}^{\alpha_2}$;*
2. *if $\alpha_1(\bar{\mathbf{z}}) + \alpha_2(\bar{\mathbf{z}}) - a \leq \beta_1(\bar{\mathbf{z}}) + \beta_2(\bar{\mathbf{z}}) - b$, then a subgradient of $g^{\max}$ at $\bar{\mathbf{z}}$ is given by $\mathbf{s}^{\beta_1} + \mathbf{s}^{\beta_2}$.*

Similarly, suppose that $\gamma_1, \gamma_2, \delta_1, \delta_2 : Z \to \mathbb{R}$ are concave, $\mathbf{s}^{\gamma_1}$ is a subgradient of γ_1 at $\bar{\mathbf{z}} \in Z$, $\mathbf{s}^{\gamma_2}$ is a subgradient of γ_2 at $\bar{\mathbf{z}}$, $\mathbf{s}^{\delta_1}$ is a subgradient of δ_1 at $\bar{\mathbf{z}}$, $\mathbf{s}^{\delta_2}$ is a subgradient of δ_2 at $\bar{\mathbf{z}}$, and $c, d \in \mathbb{R}$ are constants. Then, the function $g^{\min} : Z \to \mathbb{R}$

$$g^{\min}(\mathbf{z}) \equiv \min\{\gamma_1(\mathbf{z}) + \gamma_2(\mathbf{z}) - c, \delta_1(\mathbf{z}) + \delta_2(\mathbf{z}) - d\}$$

is concave on Z and

1. *if $\gamma_1(\bar{\mathbf{z}}) + \gamma_2(\bar{\mathbf{z}}) - c \leq \delta_1(\bar{\mathbf{z}}) + \delta_2(\bar{\mathbf{z}}) - d$, then a subgradient of $g^{\min}$ at $\bar{\mathbf{z}}$ is given by $\mathbf{s}^{\gamma_1} + \mathbf{s}^{\gamma_2}$;*
2. *if $\gamma_1(\bar{\mathbf{z}}) + \gamma_2(\bar{\mathbf{z}}) - c \geq \delta_1(\bar{\mathbf{z}}) + \delta_2(\bar{\mathbf{z}}) - d$, then a subgradient of $g^{\min}$ at $\bar{\mathbf{z}}$ is given by $\mathbf{s}^{\delta_1} + \mathbf{s}^{\delta_2}$.*

For the following proposition, see also Theorem D-4.1.1 from Hiriart-Urruty and Lemaréchal [16].

PROPOSITION 2.11 (scalar multiplication rule for subgradients). *Suppose that $Z \subset \mathbb{R}^n$ is a nonempty convex set, $f^u : Z \to \mathbb{R}$ is a convex function, $\mathbf{s}^u$ is a subgradient of f^u at $\bar{\mathbf{z}} \in Z$, $f^o : Z \to \mathbb{R}$ is a concave function, $\mathbf{s}^o$ is a subgradient of f^o at $\bar{\mathbf{z}}$, and $\kappa \in \mathbb{R}$ is a constant. Then, the function $\alpha : Z \to \mathbb{R}$,*

$$\alpha(\mathbf{z}) = \begin{cases} \kappa f^u(\mathbf{z}) & \text{if } \kappa \geq 0, \\ \kappa f^o(\mathbf{z}) & \text{otherwise,} \end{cases}$$

is convex on Z and

1. *if $\kappa \geq 0$, then $\kappa\mathbf{s}^u$ is a subgradient of α at $\bar{\mathbf{z}}$;*
2. *if $\kappa < 0$, then $\kappa\mathbf{s}^o$ is a subgradient of α at $\bar{\mathbf{z}}$.*

Similarly, the function $\beta : Z \to \mathbb{R}$

$$\beta(\mathbf{z}) = \begin{cases} \kappa f^o(\mathbf{z}) & \text{if } \kappa \geq 0, \\ \kappa f^u(\mathbf{z}) & \text{otherwise,} \end{cases}$$

is concave on Z and

1. *if $\kappa \geq 0$, then $\kappa\mathbf{s}^o$ is a subgradient of β at $\bar{\mathbf{z}}$;*
2. *if $\kappa < 0$, then $\kappa\mathbf{s}^u$ is a subgradient of β at $\bar{\mathbf{z}}$.*

The following lemma is based on Theorem D-4.3.1 from Hiriart-Urruty and Lemaréchal [16].

LEMMA 2.12 (subgradient rule for postcomposition with a function). *Let $X \subset \mathbb{R}$ and $Z \subset \mathbb{R}^n$ be nonempty convex sets and $\bar{\mathbf{z}} \in Z$. Suppose that $F^u, F^o : X \to \mathbb{R}$ are, respectively, a convex and concave function, and $f^u, f^o : Z \to \mathbb{R}$ are, respectively, a convex and concave function, with $f^u(Z) \subset X$ and $f^o(Z) \subset X$. Consider $x^u, x^o \in X$ such that $x^u = f^u(\bar{\mathbf{z}})$ and $x^o = f^o(\bar{\mathbf{z}})$. Let σ^{uu} be a subgradient of F^u at x^u, σ^{uo} be a subgradient of F^u at x^o, σ^{ou} be a subgradient of F^o at x^u, σ^{oo} be a subgradient of F^o at x^o, $\mathbf{s}^u$ be a subgradient of f^u at $\bar{\mathbf{z}}$, and $\mathbf{s}^o$ be a subgradient of f^o at $\bar{\mathbf{z}}$.*

1. *Suppose that F^u is nondecreasing. Then, the composite function $F^u \circ f^u$ is convex on Z, and $\sigma^{uu}\mathbf{s}^u$ is a subgradient of $F^u \circ f^u$ at $\bar{\mathbf{z}}$.*
2. *Suppose that F^u is nonincreasing. Then, the composite function $F^u \circ f^o$ is convex on Z, and $\sigma^{uo}\mathbf{s}^o$ is a subgradient of $F^u \circ f^o$ at $\bar{\mathbf{z}}$.*
3. *S*uppose that F^o is nonincreasing. Then, the composite function $F^o \circ f^u$ is concave on Z, and $\sigma^{ou}\mathbf{s}^u$ is a subgradient of $F^o \circ f^u$ at $\bar{\mathbf{z}}$.

4. *S*uppose that F^o is nondecreasing. Then, the composite function $F^o \circ f^o$ is concave on Z, and $\sigma^{oo}\mathbf{s}^o$ is a subgradient of $F^o \circ f^o$ at $\bar{\mathbf{z}}$.

3. Subgradient propagation.

In this section, the development and implementation of the recursive subgradient propagation through the McCormick relaxations are presented.

3.1. Theoretical development.

In order to employ Lemma 2.12 for the calculation of subgradients of McCormick relaxations, the following intermediate result will be used.

LEMMA 3.1. *Let $X \subset \mathbb{R}$ be a nonempty convex set.*

1. *Let $f^u : X \to \mathbb{R}$ be convex. Suppose that the infimum of f^u on X is attained, and $x^{\min} \in \arg\min_{x\in X} f^u(x)$. Then*
   (a) *f^u is nonincreasing for $x \leq x^{\min}$,*
   (b) *f^u is nondecreasing for $x \geq x^{\min}$.*
2. *Similarly, let $f^o : X \to \mathbb{R}$ be concave. Suppose that the supremum of f^o on X is attained, and $x^{\max} \in \arg\max_{x\in X} f^o(x)$. Then*
   (a) *f^o is nondecreasing for $x \leq x^{\max}$,*
   (b) *f^o is nonincreasing for $x \geq x^{\max}$.*

The proof is elementary.

Subgradient calculation for the composition theorem by McCormick is now presented. Note that by construction $f^u(\mathbf{z}) \leq f^o(\mathbf{z})$ for all $\mathbf{z} \in Z$, and therefore, the three cases considered below cover all possibilities.

THEOREM 3.2 (subgradients from McCormick's composition). *Suppose that $Z \subset \mathbb{R}^n$ is a nonempty convex set. Suppose further that Theorem 2.8 has been applied to the composite function $F \circ f$ to obtain the convex relaxation $g^u : Z \to \mathbb{R}$,*

$$g^u(\mathbf{z}) = F^u(\mathrm{mid}\{f^u(\mathbf{z}), f^o(\mathbf{z}), x^{\min}\})$$

and the concave relaxation $g^o : Z \to \mathbb{R}$,

$$g^o(\mathbf{z}) = F^o(\mathrm{mid}\{f^u(\mathbf{z}), f^o(\mathbf{z}), x^{\max}\}).$$

Consider $\bar{\mathbf{z}} \in Z$ and $x^u, x^o \in X$ such that $x^u = f^u(\bar{\mathbf{z}})$ and $x^o = f^o(\bar{\mathbf{z}})$. Suppose that σ^{uu} is a subgradient of F^u at x^u, σ^{uo} is a subgradient of F^u at x^o, $\mathbf{s}^u$ is a subgradient of f^u at $\bar{\mathbf{z}}$, σ^{ou} is a subgradient of F^o at x^u, σ^{oo} is a subgradient of F^o at x^o, and $\mathbf{s}^o$ is a subgradient of f^o at $\bar{\mathbf{z}}$. Then, a subgradient of g^u at $\bar{\mathbf{z}}$ is given by

1. $\mathbf{0}$ *if $f^u(\bar{\mathbf{z}}) \leq x^{\min} \leq f^o(\bar{\mathbf{z}})$,*
2. $\sigma^{uo}\mathbf{s}^o$ *if $f^o(\bar{\mathbf{z}}) < x^{\min}$,*
3. $\sigma^{uu}\mathbf{s}^u$ *if $x^{\min} < f^u(\bar{\mathbf{z}})$,*

and a subgradient of g^o at $\bar{\mathbf{z}}$ is given by

1. $\mathbf{0}$ *if $f^u(\bar{\mathbf{z}}) \leq x^{\max} \leq f^o(\bar{\mathbf{z}})$,*
2. $\sigma^{oo}\mathbf{s}^o$ *if $f^o(\bar{\mathbf{z}}) < x^{\max}$,*
3. $\sigma^{ou}\mathbf{s}^u$ *if $x^{\max} < f^u(\bar{\mathbf{z}})$.*

Proof. From the McCormick construction, the function $\mathrm{mid}\{f^u(\mathbf{z}), f^o(\mathbf{z}), x^{\min}\} \in X\ \forall \mathbf{z} \in Z$.

1. If $f^u(\bar{\mathbf{z}}) \leq x^{\min} \leq f^o(\bar{\mathbf{z}})$, then $F^u(\mathrm{mid}\{f^u(\bar{\mathbf{z}}), f^o(\bar{\mathbf{z}}), x^{\min}\}) = F^u(x^{\min})$, and the result follows immediately because

$$F^u(x^{\min}) \leq F^u(\mathrm{mid}\{f^u(\mathbf{z}), f^o(\mathbf{z}), x^{\min}\}) \quad \text{for all } \mathbf{z} \in Z$$

by the definition of $x^{\min}$, since $\mathrm{mid}\{f^u(\mathbf{z}), f^o(\mathbf{z}), x^{\min}\} \in X$.

2. If $f^o(\bar{\mathbf{z}}) < x^{\min}$, then from Lemma 3.1 part 1a, there exists a neighborhood of $f^o(\bar{\mathbf{z}})$ on which F^u is nonincreasing. Moreover, $F^u(\text{mid}\{f^u(\bar{\mathbf{z}}), f^o(\bar{\mathbf{z}}), x^{\min}\}) = F^u(f^o(\bar{\mathbf{z}}))$. Therefore, the hypotheses of Lemma 2.12 part 2 are satisfied, and $\sigma^{uo}\mathbf{s}^o$ is a subgradient of g^u at $\bar{\mathbf{z}}$.
3. If $x^{\min} < f^u(\bar{\mathbf{z}})$, then from Lemma 3.1 part 1b there exists a neighborhood of $f^u(\bar{\mathbf{z}})$ on which F^u is nondecreasing. Moreover, $F^u(\text{mid}\{f^u(\bar{\mathbf{z}}), f^o(\bar{\mathbf{z}}), x^{\min}\}) = F^u(f^u(\bar{\mathbf{z}}))$. Therefore, the hypotheses of Lemma 2.12 part 1 are satisfied, and $\sigma^{uu}\mathbf{s}^u$ is a subgradient of g^u at $\bar{\mathbf{z}}$.

The proof for the subgradients of g^o is analogous. $\square$

Note that in parts 2 and 3 in the above proof, Lemma 2.12 is only used for a neighborhood; this is justified by an examination of the proof of Lemma 2.12.

A rule is also needed for the subgradients of the relaxation of a product of functions.

THEOREM 3.3 (multiplication rule for subgradients). *Suppose that $Z \subset \mathbb{R}^n$ is a nonempty convex set, and $\bar{\mathbf{z}} \in Z$. Let $g, g_1, g_2 : Z \to \mathbb{R}$ such that $g(\mathbf{z}) = g_1(\mathbf{z})g_2(\mathbf{z})$. Let $g_1^u, g_1^o : Z \to \mathbb{R}$ be a convex and concave relaxation of g_1 on Z, respectively. Similarly, let $g_2^u, g_2^o : Z \to \mathbb{R}$ be a convex and concave relaxation of g_2 on Z, respectively. Furthermore, let $g_1^L, g_1^U, g_2^L, g_2^U \in \mathbb{R}$ such that*

$$g_1^L \le g_1(\mathbf{z}) \le g_1^U \quad \textit{for all } \mathbf{z} \in Z \quad \textit{and} \quad g_2^L \le g_2(\mathbf{z}) \le g_2^U \quad \textit{for all } \mathbf{z} \in Z.$$

Consider the following intermediate functions, $\alpha_1, \alpha_2, \beta_1, \beta_2, \gamma_1, \gamma_2, \delta_1, \delta_2 : Z \to \mathbb{R}$:

$$\begin{aligned}
\alpha_1(\mathbf{z}) &= \min\left\{g_2^L g_1^u(\mathbf{z}), g_2^L g_1^o(\mathbf{z})\right\}, & \alpha_2(\mathbf{z}) &= \min\left\{g_1^L g_2^u(\mathbf{z}), g_1^L g_2^o(\mathbf{z})\right\},\\
\beta_1(\mathbf{z}) &= \min\left\{g_2^U g_1^u(\mathbf{z}), g_2^U g_1^o(\mathbf{z})\right\}, & \beta_2(\mathbf{z}) &= \min\left\{g_1^U g_2^u(\mathbf{z}), g_1^U g_2^o(\mathbf{z})\right\},\\
\gamma_1(\mathbf{z}) &= \max\left\{g_2^L g_1^u(\mathbf{z}), g_2^L g_1^o(\mathbf{z})\right\}, & \gamma_2(\mathbf{z}) &= \max\left\{g_1^U g_2^u(\mathbf{z}), g_1^U g_2^o(\mathbf{z})\right\},\\
\delta_1(\mathbf{z}) &= \max\left\{g_2^U g_1^u(\mathbf{z}), g_2^U g_1^o(\mathbf{z})\right\}, & \delta_2(\mathbf{z}) &= \max\left\{g_1^L g_2^u(\mathbf{z}), g_1^L g_2^o(\mathbf{z})\right\}.
\end{aligned}$$

Then, α_1, α_2, β_1, and β_2 are convex on Z, and γ_1, γ_2, δ_1, and δ_2 are concave on Z. The functions $g^u, g^o : Z \to \mathbb{R}$, such that

$$\begin{aligned}
g^u(\mathbf{z}) &= \max\left\{\alpha_1(\mathbf{z}) + \alpha_2(\mathbf{z}) - g_1^L g_2^L, \beta_1(\mathbf{z}) + \beta_2(\mathbf{z}) - g_1^U g_2^U\right\},\\
g^o(\mathbf{z}) &= \min\left\{\gamma_1(\mathbf{z}) + \gamma_2(\mathbf{z}) - g_1^U g_2^L, \delta_1(\mathbf{z}) + \delta_2(\mathbf{z}) - g_1^L g_2^U\right\},
\end{aligned}$$

are, respectively, a convex and concave relaxation of g on Z.

Moreover, subgradients of α_1, α_2, β_1, β_2, γ_1, γ_2, δ_1, δ_2 at $\bar{\mathbf{z}}$ are given by

$$\mathbf{s}^{\alpha_1} = \begin{cases} g_2^L \mathbf{s}^{g_1^u} & \textit{if } g_2^L \ge 0,\\ g_2^L \mathbf{s}^{g_1^o} & \textit{otherwise,}\end{cases} \qquad \mathbf{s}^{\alpha_2} = \begin{cases} g_1^L \mathbf{s}^{g_2^u} & \textit{if } g_1^L \ge 0,\\ g_1^L \mathbf{s}^{g_2^o} & \textit{otherwise,}\end{cases}$$

$$\mathbf{s}^{\beta_1} = \begin{cases} g_2^U \mathbf{s}^{g_1^u} & \textit{if } g_2^U \ge 0,\\ g_2^U \mathbf{s}^{g_1^o} & \textit{otherwise,}\end{cases} \qquad \mathbf{s}^{\beta_2} = \begin{cases} g_1^U \mathbf{s}^{g_2^u} & \textit{if } g_1^U \ge 0,\\ g_1^U \mathbf{s}^{g_2^o} & \textit{otherwise,}\end{cases}$$

$$\mathbf{s}^{\gamma_1} = \begin{cases} g_2^L \mathbf{s}^{g_1^o} & \textit{if } g_2^L \ge 0,\\ g_2^L \mathbf{s}^{g_1^u} & \textit{otherwise,}\end{cases} \qquad \mathbf{s}^{\gamma_2} = \begin{cases} g_1^U \mathbf{s}^{g_2^o} & \textit{if } g_1^U \ge 0,\\ g_1^U \mathbf{s}^{g_2^u} & \textit{otherwise,}\end{cases}$$

$$\mathbf{s}^{\delta_1} = \begin{cases} g_2^U \mathbf{s}^{g_1^o} & \textit{if } g_2^U \ge 0,\\ g_2^U \mathbf{s}^{g_1^u} & \textit{otherwise,}\end{cases} \qquad \mathbf{s}^{\delta_2} = \begin{cases} g_1^L \mathbf{s}^{g_2^o} & \textit{if } g_1^L \ge 0,\\ g_1^L \mathbf{s}^{g_2^u} & \textit{otherwise,}\end{cases}$$

where $\mathbf{s}^{g_1^u}$, $\mathbf{s}^{g_1^o}$, $\mathbf{s}^{g_2^u}$, $\mathbf{s}^{g_2^o}$ are, respectively, subgradients of g_1^u, g_1^o, g_2^u, g_2^o at $\bar{\mathbf{z}}$.

Finally, a subgradient of g^u at $\bar{\mathbf{z}}$ is given by
1. $\mathbf{s}^{\alpha_1} + \mathbf{s}^{\alpha_2}$ *if* $\alpha_1(\bar{\mathbf{z}}) + \alpha_2(\bar{\mathbf{z}}) - a \geq \beta_1(\bar{\mathbf{z}}) + \beta_2(\bar{\mathbf{z}}) - b$,
2. $\mathbf{s}^{\beta_1} + \mathbf{s}^{\beta_2}$ *if* $\alpha_1(\bar{\mathbf{z}}) + \alpha_2(\bar{\mathbf{z}}) - a \leq \beta_1(\bar{\mathbf{z}}) + \beta_2(\bar{\mathbf{z}}) - b$,

and a subgradient of g^o at $\bar{\mathbf{z}}$ is given by
1. $\mathbf{s}^{\gamma_1} + \mathbf{s}^{\gamma_2}$ *if* $\gamma_1(\bar{\mathbf{z}}) + \gamma_2(\bar{\mathbf{z}}) - c \leq \delta_1(\bar{\mathbf{z}}) + \delta_2(\bar{\mathbf{z}}) - d$,
2. $\mathbf{s}^{\delta_1} + \mathbf{s}^{\delta_2}$ *if* $\gamma_1(\bar{\mathbf{z}}) + \gamma_2(\bar{\mathbf{z}}) - c \geq \delta_1(\bar{\mathbf{z}}) + \delta_2(\bar{\mathbf{z}}) - d$.

Proof. Recall that by Proposition 2.7, g^u and g^o are, respectively, a convex and concave relaxation of g on Z. The calculation of subgradients for α_1, α_2, β_1, β_2, γ_1, γ_2, δ_1, δ_2 follows directly from their definition and the properties of the min and max functions. Finally, by Proposition 2.7, α_1, α_2, β_1, and β_2 are convex, and Proposition 2.10 can be invoked to show the desired result for the subgradient of g^u. Similarly, by Proposition 2.7, γ_1, γ_2, δ_1, and δ_2 are concave, and Proposition 2.10 can be again invoked to show the desired result for the subgradient of g^o. $\square$

3.2. Application. In the following, it is briefly discussed how to employ the above presented results for the systematic propagation of subgradients of the convex and concave relaxations of factorable functions. Similar to the McCormick relaxations, a bounded convex set $Z \subset \mathbb{R}^n$ is assumed along with a real-valued factorable function $g : Z \to \mathbb{R}$, that can be represented in such a way that all univariate functions have known convex and concave relaxations as well as known enclosures for their ranges. An implicit assumption in the construction of McCormick relaxations is that the enclosure propagation is sufficiently tight that there is no domain violation for any function. As Example 3.1 shows, in some cases additional information is required to construct the relaxations.

Example 3.1 (domain violation). Consider $Z = [-1, 1]$ and the factorable function $g : Z \to \mathbb{R}$, $g(z) = \sqrt{|z| + z^3}$. Let $f : Z \to \mathbb{R}$, $f(z) = |z| + z^3$, and note $f(Z) = [0, 2]$. The natural interval extension of $|z| + z^3$ on Z gives $[0, 1] + [-1, 1] = [-1, 2]$, and the square root is not defined on this domain.

In the following, functions are assumed for which McCormick relaxations can be constructed. Recall that existence of subgradients on the interior of Z is guaranteed [4]. In order to calculate a subgradient at an interior point for a McCormick relaxation, the only additional requirement is that subgradients for the relaxations of the univariate functions can be calculated for any interior point of their domain. The combination of Proposition 2.9 and Theorems 3.2 and 3.3 provides a simple extension to the McCormick relaxations that allows the subgradient propagation for any $\bar{\mathbf{z}} \in \text{int}(Z)$ without any further assumptions. For sums and products, the subgradients of the constituent functions are simply used to calculate a subgradient of the composite function. The following proposition shows that in the composition $F \circ f$ of Theorem 3.2, the subgradients of the relaxations of the univariate function are only required for points in the interior of the domain of the univariate function. The trivial case that the inner function is a constant is excluded.

PROPOSITION 3.4. *Consider the application of Theorem* 3.2 *for* $\bar{\mathbf{z}} \in \text{int}(Z)$ *and assume that f is not a constant on Z.*
1. *If* $f^o(\bar{\mathbf{z}}) < x^{\min}$, *then* $f^o(\bar{\mathbf{z}}) \in \text{int}(X)$.
2. *If* $x^{\min} < f^u(\bar{\mathbf{z}})$, *then* $f^u(\bar{\mathbf{z}}) \in \text{int}(X)$.
3. *If* $f^o(\bar{\mathbf{z}}) < x^{\max}$, *then* $f^o(\bar{\mathbf{z}}) \in \text{int}(X)$.
4. *If* $x^{\max} < f^u(\bar{\mathbf{z}})$, *then* $f^u(\bar{\mathbf{z}}) \in \text{int}(X)$.

As a consequence, subgradients of the univariate relaxations F^u, F^o are only required on $\text{int}(X)$.

Proof. Only the first of the four cases is proved; the other three cases are analogous. Let $X = [x^L, x^U]$. Since $x^{\min} \in [x^L, x^U]$ by $f^o(\bar{\mathbf{z}}) < x^{\min}$, it follows $f^o(\bar{\mathbf{z}}) < x^U$. It now remains to show $x^L < f^o(\bar{\mathbf{z}})$.

Since f^o is concave on Z and $\bar{\mathbf{z}} \in \text{int}(Z)$, there exists $\mathbf{s}^o$ such that for all $\mathbf{z} \in Z$

$$f^o(\mathbf{z}) \leq f^o(\bar{\mathbf{z}}) + (\mathbf{s}^o)^{\mathrm{T}}(\mathbf{z} - \bar{\mathbf{z}}).$$

Since $\bar{\mathbf{z}} \in \text{int}(Z)$, there exists $\delta > 0$ such that $N_\delta(\bar{\mathbf{z}}) \subset Z$, where $N_\delta(\bar{\mathbf{z}})$ denotes an open neighborhood of $\bar{\mathbf{z}}$ with radius δ.

Suppose first that $\mathbf{s}^o \neq \mathbf{0}$, i.e., there exists $i \in \{1, \ldots, n\}$ such that $s_i^o \neq 0$. Take $\hat{\mathbf{z}} \in Z$ such that

$$\hat{z}_i = \bar{z}_i - \text{sign}(s_i)\,\delta/2 \qquad \text{and} \qquad \hat{z}_j = \bar{z}_j \quad \forall j \in \{1, \ldots, n\} : j \neq i.$$

Clearly, $f^o(\hat{\mathbf{z}}) < f^o(\bar{\mathbf{z}})$. Since f^o is an overestimator of f on Z, it follows $f(\hat{\mathbf{z}}) \leq f^o(\hat{\mathbf{z}})$. Moreover, $x^L \leq f(\hat{\mathbf{z}})$ by the definition of x^L. Combining the last three inequalities, $x^L < f^o(\bar{\mathbf{z}})$ is obtained.

Suppose now that $\mathbf{s}^o = \mathbf{0}$. Then, for all $\mathbf{z} \in Z$, it follows $f^o(\mathbf{z}) \leq f^o(\bar{\mathbf{z}})$, which together with $f(\mathbf{z}) \leq f^o(\mathbf{z})$, gives

$$f(\mathbf{z}) \leq f^o(\mathbf{z}) \leq f^o(\bar{\mathbf{z}}) \qquad \forall \mathbf{z} \in Z.$$

By definition $x^L \leq f(\mathbf{z})$ for all $\mathbf{z} \in Z$. Since by assumption $f(Z) \neq \{x^L\}$, there exists $\hat{\mathbf{z}} \in Z$ such that (s.t.) $x^L < f(\hat{\mathbf{z}})$. Therefore, $x^L < f(\hat{\mathbf{z}}) \leq f^o(\bar{\mathbf{z}})$.

Recall that by Theorem 3.2, a subgradient to g^u at $\bar{\mathbf{z}}$ is given by

1. $\mathbf{0}$ if $f^u(\bar{\mathbf{z}}) \leq x^{\min} \leq f^o(\bar{\mathbf{z}})$,
2. $\sigma^{uo}\mathbf{s}^o$ if $f^o(\bar{\mathbf{z}}) < x^{\min}$,
3. $\sigma^{uu}\mathbf{s}^u$ if $x^{\min} < f^u(\bar{\mathbf{z}})$.

In the first case, the subgradients of F^u are not used. In the second case, $f^o(\bar{\mathbf{z}}) \in \text{int}(X)$, and therefore, σ^{uo} is a subgradient of F^u at a point in the interior of X. The third case is analogous to the second. The subgradients of g^o are also analogous. ∎

The calculation of subgradients at boundary points of Z may also be of interest. For this case some additional assumptions are necessary. For instance, the square root is a concave function with domain $[0, +\infty)$ which is differentiable on $(0, +\infty)$, but at 0, no subgradient exists. A sufficient condition is that all the convex and concave relaxations of univariate functions are differentiable on their domains.

3.2.1. Affine relaxations. By the definition of the subgradient, it is clear that with the usual notation and for some $\bar{\mathbf{z}} \in Z$, the functions $f^{ul}, f^{ol} : Z \to \mathbb{R}$, such that

$$f^{ul}(\mathbf{z}) \equiv f^u(\bar{\mathbf{z}}) + (\mathbf{s}^u)^{\mathrm{T}}(\mathbf{z} - \bar{\mathbf{z}}) \qquad \text{and} \qquad f^{ol}(\mathbf{z}) \equiv f^o(\bar{\mathbf{z}}) + (\mathbf{s}^o)^{\mathrm{T}}(\mathbf{z} - \bar{\mathbf{z}}),$$

are, respectively, an affine underestimator and an affine overestimator of f on Z. As described in the introduction, the calculation of these affine relaxations is one of the goals of this paper. Similarly, to affine relaxations of smooth functions, the choice of $\bar{\mathbf{z}}$ greatly affects how closely f^{ul} and f^{ol} approximate the original function. Moreover, in the case where f^u and/or f^o are not differentiable at $\bar{\mathbf{z}}$, the choice of subgradient also affects the approximation.

This linearization allows the calculation of computationally inexpensive bounds for nonconvex optimization problems. Consider, for instance, a box-constrained problem $\min_{\mathbf{z} \in [\mathbf{z}^L, \mathbf{z}^U]} f(\mathbf{z})$, with the convex relaxation $\min_{\mathbf{z} \in [\mathbf{z}^L, \mathbf{z}^U]} f^u(\mathbf{z})$. Pick an arbitrary point $\bar{\mathbf{z}}$ in the interior of $Z$and construct the further relaxation

$$f^{ul,*} = \min_{\mathbf{z} \in [\mathbf{z}^L, \mathbf{z}^U]} f^{ul}(\mathbf{z}),$$

which is a box-constrained problem with an affine objective function. Therefore, an optimal solution point $\mathbf{z}^{ul,*}$ (nonunique in general) is given by

$$z_j^{ul,*} = \begin{cases} z_j^L & \text{if } s_j^u \leq 0, \\ z_j^U & \text{otherwise,} \end{cases} \qquad j = 1, \ldots, n.$$

The optimal solution value of the affine relaxation (lower bound to the nonconvex optimization problem) is given by $f^{ul}(\mathbf{z}^{ul,*})$ with a small effort in addition to the computation of a subgradient. This procedure is much cheaper than the solution of the convex relaxation and comparable to evaluation of the natural interval extension.

Suppose further that the linearization is performed at a finite number of points $\bar{\mathbf{z}}^i$. The above relaxations allow the calculation of an optimal solution point and the optimal solution value for each of a collection of box-constrained LPs

$$f^{ul,i,*} = \min_{\mathbf{z} \in [\mathbf{z}^L, \mathbf{z}^U]} f^{ul,i}(\mathbf{z}).$$

A valid lower bound to the nonconvex optimization problem is then given by $\max_i f^{ul,i,*}$. Clearly, the choice of $\bar{\mathbf{z}}^i$ affects the tightness of the relaxations. Choosing good points $\bar{\mathbf{z}}^i$ is outside the scope of this article.

The construction of upper bounds to $\max_{\mathbf{z} \in [\mathbf{z}^L, \mathbf{z}^U]} f(\mathbf{z})$ is analogous by constructing an affine relaxation $\max_{\mathbf{z} \in [\mathbf{z}^L, \mathbf{z}^U]} f^{ol}(\mathbf{z})$. For general constraints, it is still possible to calculate cheap lower bounds via a relaxation of the feasible set.

3.3. Illustrative example. In this subsection, the propagation of subgradients is given for a simple function.

Example 3.2 (simple composite function). Let $Z = [-1, 1]^2$, $\bar{\mathbf{z}} = \mathbf{0}$, $f : Z \to \mathbb{R}$ such that $f(\mathbf{z}) = z_1 + |z_2|$. Let $X = [-1, 2]$ and $F : X \to \mathbb{R}$ such that $F(x) = x^2$. An affine underestimator and an affine overestimator of $g = F \circ f$, $g(\mathbf{z}) = (z_1 + |z_2|)^2$ are constructed by linearizing the convex and concave relaxations obtained by Theorem 2.8 at $\bar{\mathbf{z}} = \mathbf{0}$. These are shown in Figure 3.1. Note that neither f nor g is differentiable at $\bar{\mathbf{z}}$.

The inner function f is convex on Z, and therefore, $f^u : Z \to \mathbb{R}$ such that $f^u(\mathbf{z}) = z_1 + |z_2|$ is a convex relaxation of f on Z, and $f^o : Z \to \mathbb{R}$ such that $f^o(\mathbf{z}) = z_1 + 1$ is a concave relaxation of f on Z. A subgradient of f^u at $\mathbf{0}$ is given by $\mathbf{s}^u = (1, 0)$. Since f^o is differentiable, its unique subgradient at $\mathbf{0}$ is given by $\mathbf{s}^o = (1, 0)$. Since $f(Z) = [-1, 2]$, it follows $f(Z) \subset X$.

The outer function F is convex on X, and therefore, $F^u : X \to \mathbb{R}$ such that $F^u(x) = x^2$ is a convex relaxation of F on X, and $F^o : X \to \mathbb{R}$ such that $F^o(x) = 2 + x$ is a concave relaxation of F on X. Note that F^u and F^o are, respectively, the convex and concave envelope of F on Z. Moreover, $x^{\min} = 0$ is the unique minimum of F^u on X, and $x^{\max} = 2$ is the unique maximum of F^o on X. Since F^u is differentiable, its unique subgradient is given by the derivative $2x$. Similarly, the unique subgradient of F^o is 1.

By Theorem 2.8, $g^u : Z \to \mathbb{R}$ such that

$$g^u(\mathbf{z}) = F^u\left(\text{mid}\left\{f^u(\mathbf{z}), f^o(\mathbf{z}), x^{\min}\right\}\right) = \left(\text{mid}\left\{z_1 + |z_2|, z_1 + 1, 0\right\}\right)^2$$

is a convex relaxation of g. Since $z_1 + 1 \geq 0$ for all $\mathbf{z} \in Z$, it follows $\text{mid}\{z_1 + |z_2|, z_1 + 1, 0\} = \max\{z_1 + |z_2|, 0\}$, and therefore, $g^u(\mathbf{z}) = (\max\{z_1 + |z_2|, 0\})^2$. Thus, $0 = f^u(\mathbf{0}) \leq x^{\min} \leq f^o(\mathbf{0}) = 1$, and therefore, by Theorem 3.2, a subgradient of g^u at $\mathbf{0}$ is given by $\mathbf{0}$.

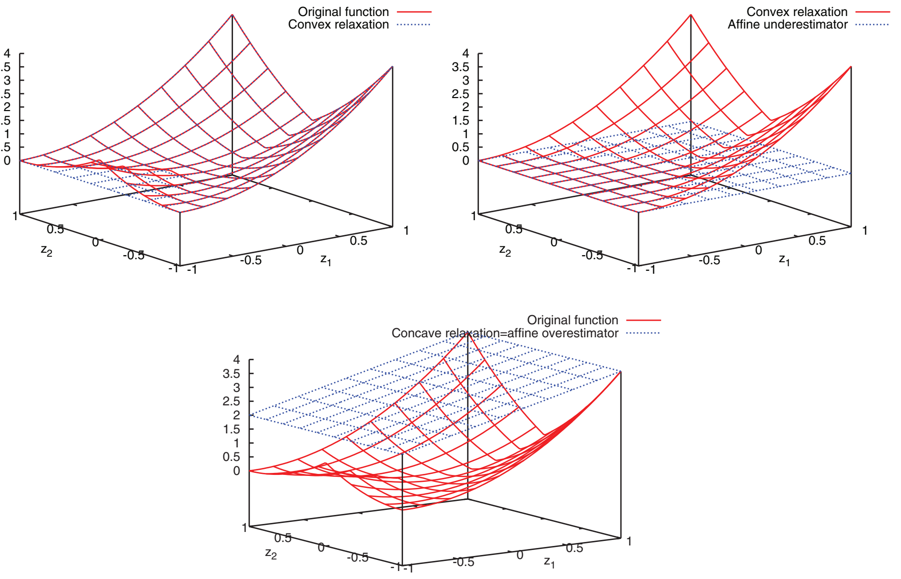


FIG. 3.1. *Graphical illustration of Example* 3.2.

By Theorem 2.8, $g^o : Z \to \mathbb{R}$ such that

$$g^o(\mathbf{z}) = F^u\left(\operatorname{mid}\left\{f^u(\mathbf{z}), f^o(\mathbf{z}), x^{\max}\right\}\right) = 2 + \operatorname{mid}\left\{z_1 + |z_2|, z_1 + 1, 2\right\}$$

is a concave relaxation of g. Since $z_1 + |z_2| \leq z_1 + 1 \leq 2$ for all $\mathbf{z} \in Z$, it follows $\operatorname{mid}\{z_1 + |z_2|, z_1 + 1, 2\} = z_1 + 1$, and therefore, $g^o(\mathbf{z}) = 3 + z_1$. In particular, $1 = f^o(\mathbf{0}) < x^{\max} = 2$, and therefore, by Theorem 3.2, a subgradient of g^o at $\mathbf{0}$ is given by $1 \cdot (1, 0) = (1, 0)$.

4. Implementation.

The application of the theory presented in section 3 is both tedious and error-prone for all but the simplest problems, thus, making manual derivation of the relaxations and their subgradients difficult. Fortunately, the theory lends itself naturally to automation by a computer.

4.1. Background on automatic differentiation.

AD [13] is a method to evaluate numerically the derivative of a function specified by a computer program. It exploits the fact that any computer program that implements a function, say, $y = f(\mathbf{x})$, $\mathbf{x} \in \mathbb{R}^n$, can be decomposed into a finite sequence of elementary (unary and binary) operations, any one of which may be differentiated according to the basic rules of calculus. These elementary derivatives are then combined in accordance with the chain rule to evaluate some derivative information for f (such as tangents, gradients, or the Jacobian matrix).

AD is most easily accomplished by the so-called forward mode, which augments the algebra of real numbers to obtain a new arithmetic. Suppose, for simplicity, that we want to differentiate the output variable y with respect to the first element x_1. To the result v of an elementary operation, a number $\dot{v} \equiv \frac{\partial v}{\partial x_1}$ is augmented, which represents the numerical value of the derivative of that variable with respect to x_1. The basic unary and binary arithmetic operators for a pair $(v, \dot{v})$ in the augmented algebra can be extended by considering ordinary arithmetic for v, and first-order

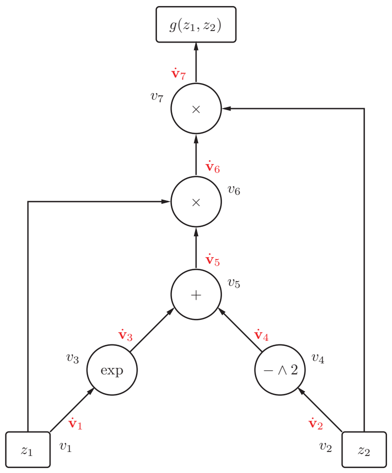


FIG. 4.1. *Computational graph for forward AD in Example* 4.1.

differentiation arithmetic for $\dot{v}$, e.g.,

$$\begin{aligned}(v,\dot{v})+(w,\dot{w})&=(v+w,\dot{v}+\dot{w}),\\ (v,\dot{v})\cdot(w,\dot{w})&=(vw,\dot{v}w+v\dot{w}),\\ f((v,\dot{v}))&=(f(v),f'(v)\dot{v}).\end{aligned}$$

Clearly, the same procedure can be used to evaluate the derivative of y with respect to the other variables $x_2,\ldots,x_n$. In particular, the elemental derivatives $\dot{v}$ can be redefined to be vectors (rather than scalars) so as to evaluate several partial derivatives at once. These considerations are illustrated by the following example.

Example 4.1 (forward mode of AD). Let $Z=[-1,3]\times[-2,3]$ and $g:Z\to\mathbb{R}$ such that $g(\mathbf{z})=(\exp(z_1)-{z_2}^2)z_1z_2$. Treating $y\mapsto -y^2$ as a univariate (concave) intrinsic function, a decomposition of g into a sequence of unary and binary operations is as follows:

$v_1=z_1$	$v_2=z_2$	$v_3=\exp(v_1)$	$v_4=-v_2^2$
$v_5=v_3+v_4$	$v_6=v_1v_5$	$v_7=v_2v_6$	

with $g(\mathbf{z})=v_7$.

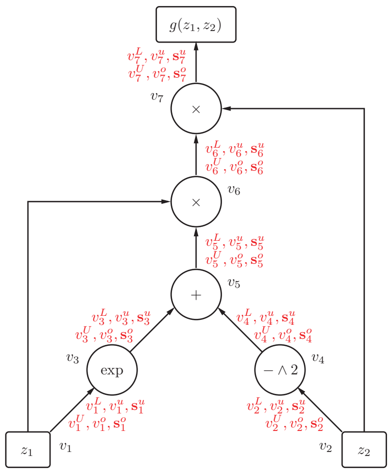
The derivatives of g are evaluated by differentiating each elementary operation in the above code list. Since both the partial derivatives $\frac{\partial g}{\partial z_1}$ and $\frac{\partial g}{\partial z_2}$ are to be calculated, the elementary derivatives $\dot{\mathbf{v}}_i$ are defined here to be vectors of size 2:

$\dot{\mathbf{v}}_1=(1,0)$	$\dot{\mathbf{v}}_2=(0,1)$	$\dot{\mathbf{v}}_3=\exp(v_1)\dot{\mathbf{v}}_1$	$\dot{\mathbf{v}}_4=-2\,v_2\,\dot{\mathbf{v}}_2$
$\dot{\mathbf{v}}_5=\dot{\mathbf{v}}_3+\dot{\mathbf{v}}_4$	$\dot{\mathbf{v}}_6=\dot{\mathbf{v}}_1v_5+v_1\dot{\mathbf{v}}_5$	$\dot{\mathbf{v}}_7=\dot{\mathbf{v}}_2v_6+v_2\dot{\mathbf{v}}_6$	

with $(\frac{\partial g}{\partial x_1},\frac{\partial g}{\partial x_2})=\dot{\mathbf{v}}_7$. This procedure evaluates exact derivatives (within rounding error).

A computational graph illustrating the evaluation of derivatives of g via the forward mode of AD is shown in Figure 4.1.

4.2. Automatic relaxation and subgradient calculation. The evaluation of convex/concave relaxations along with subgradients of these relaxations can be automated in a similar way to the forward mode of AD. Given a factorable function, a decomposition as a finite recursive composition of binary sums, binary products,



$g(z_1, z_2)$
$v_7^L, v_7^u, \mathbf{s}_7^u$
$v_7^U, v_7^o, \mathbf{s}_7^o$
v_7
$\times$
$v_6^L, v_6^u, \mathbf{s}_6^u$
$v_6^U, v_6^o, \mathbf{s}_6^o$
v_6
$\times$
$v_5^L, v_5^u, \mathbf{s}_5^u$
$v_5^U, v_5^o, \mathbf{s}_5^o$
v_5
$+$
$v_3^L, v_3^u, \mathbf{s}_3^u$
$v_3^U, v_3^o, \mathbf{s}_3^o$
$v_4^L, v_4^u, \mathbf{s}_4^u$
$v_4^U, v_4^o, \mathbf{s}_4^o$
v_3
exp
$-\wedge 2$
v_4
$v_1^L, v_1^u, \mathbf{s}_1^u$
$v_1^U, v_1^o, \mathbf{s}_1^o$
$v_2^L, v_2^u, \mathbf{s}_2^u$
$v_2^U, v_2^o, \mathbf{s}_2^o$
z_1
v_1
v_2
z_2

FIG. 4.2. *Computational graph for bound, relaxation, and subgradient propagation in Example* 4.2.

and univariate functions is first developed (see Definition 2.2), and intermediate variables are introduced for each of these elementary operations. Then, lower/upper bounds, convex/concave relaxations, and subgradients of the convex/concave relaxations are calculated for each intermediate variable, in a recursive manner. To achieve this, the algebra of real numbers is extended to a new arithmetic where 6 numbers are now augmented to every intermediate variable v, hence, leading to the septuple $(v, v^L, v^U, v^u, v^o, \mathbf{s}^u, \mathbf{s}^o)$: the numbers v^L, v^U store lower and upper bounds on the variable v; the numbers v^u, v^o store the values of the convex and concave relaxations; and the vectors $\mathbf{s}^u$, $\mathbf{s}^o$ store subgradients of the convex and concave relaxations. The arithmetic operators are extended to the augmented algebra in the following way:

- The rules to calculate lower and upper bounds for univariate functions and binary sums and products are the same as those of natural interval extensions [24].
- The rules to calculate convex and concave relaxations for univariate functions and binary sums and products are given in Theorem 2.8 and Propositions 2.6 and 2.7, respectively. Moreover, convex/concave relaxations for a variety of simple univariate functions, such as the exponential function or monomials, can be found, e.g., in [1, 32, 35, 42].
- Finally, the rules to calculate subgradients for the aforementioned convex and concave relaxations are those established earlier in Proposition 2.9 and Theorems 3.2 and 3.3.

The recursive calculation of bounds, relaxations, and subgradients is illustrated by the following example.

Example 4.2 (automatic relaxation and subgradient calculation). Consider the same function g as in Example 4.1 for which a factorable representation has already been given. The computations of bounds, relaxations, and subgradients are initialized so as to reflect the choice of independent variables z_1 and z_2, their current values, and their domain Z. We have $v_1^L = -1$, $v_1^U = 3$, $v_2^L = -2$, $v_2^U = 3$, $v_1^u = v_1^o = z_1$, $v_2^u = v_2^o = z_2$, $\mathbf{s}_1^u = \mathbf{s}_1^o = (1,0)$, and $\mathbf{s}_2^u = \mathbf{s}_2^o = (0,1)$. Then the bounds, relaxations, and subgradients are propagated as in the computational graph shown in Figure 4.2. The details of this propagation are reported in Table 4.1.

A convex relaxation and a concave relaxation of g on Z are obtained as $g^u(\mathbf{z}) = v_7^u$ and $g^o(\mathbf{z}) = v_7^o$, respectively. These relaxations are shown in the upper and lower left plots of Figure 4.3. Moreover, a subgradient of g^u and a subgradient of g^o at $\mathbf{z} \in Z$ are given by $\mathbf{s}^u = \mathbf{s}_7^u$ and $\mathbf{s}^o = \mathbf{s}_7^o$, respectively.

TABLE 4.1

$v_1^L = -1$	$v_1^U = 3$
$v_1^u = z_1$	$v_1^o = z_1$
$\mathbf{s}_1^u = (1,0)$	$\mathbf{s}_1^o = (1,0)$
$v_2^L = -2$	$v_2^U = 3$
$v_2^u = z_2$	$v_2^o = z_2$
$\mathbf{s}_2^u = (0,1)$	$\mathbf{s}_2^o = (0,1)$
$v_3^L = \exp\left(v_1^L\right)$	$v_3^U = \exp\left(v_1^U\right)$
$v_3^u = \exp\left(\operatorname{mid}\left\{v_1^u, v_1^o, v_1^L\right\}\right)$	$v_3^o = \frac{\exp\left(v_1^U\right) - \exp\left(v_1^L\right)}{v_1^U - v_1^L}\left(\operatorname{mid}\left\{v_1^u, v_1^o, v_1^U\right\} - v_1^L\right) + \exp\left(v_1^L\right)$
$x_3^{\min} = v_1^L$	$x_3^{\max} = v_1^U$
$x_3^u = v_1^u$	$x_3^o = v_1^o$
$\sigma_3^{uu} = \exp\left(x_3^u\right)$	$\sigma_3^{uo} = \exp\left(x_3^o\right)$
$\sigma_3^{ou} = \frac{\exp\left(v_1^U\right) - \exp\left(v_1^L\right)}{v_1^U - v_1^L}$	$\sigma_3^{oo} = \frac{\exp\left(v_1^U\right) - \exp\left(v_1^L\right)}{v_1^U - v_1^L}$
$\mathbf{s}_3^u = \begin{cases} \mathbf{0} & \text{if } x_3^u \leq x_3^{\min} \leq x_3^o, \\ \sigma_3^{uo}\mathbf{s}_1^o & \text{if } x_3^o < x_3^{\min}, \\ \sigma_3^{uo}\mathbf{s}_1^u & \text{otherwise.} \end{cases}$	$\mathbf{s}_3^o = \begin{cases} \mathbf{0} & \text{if } x_3^u \leq x_3^{\max} \leq x_3^o, \\ \sigma_3^{oo}\mathbf{s}_1^o & \text{if } x_3^o < x_3^{\max}, \\ \sigma_3^{ou}\mathbf{s}_1^u & \text{otherwise.} \end{cases}$
$x_4^{\min} = \begin{cases} v_2^L & \text{if } \left\lvert v_2^L\right\rvert \geq \left\lvert v_2^U\right\rvert, \\ v_2^U & \text{otherwise.} \end{cases}$	$x_4^{\max} = \operatorname{mid}\left(0, v_2^L, v_2^U\right)$
$v_4^L = -(x_4^{\min})^2$	$v_4^U = -(x_4^{\max})^2$
$v_4^u = \frac{-\left(v_2^U\right)^2 + \left(v_2^L\right)^2}{v_2^U - v_2^L}\left(\operatorname{mid}\left\{v_2^u, v_2^o, x_4^{\max}\right\} - v_2^L\right) - \left(v_2^L\right)^2$	$v_4^o = -\left(\operatorname{mid}\left\{v_2^u, v_2^o, x_4^{\min}\right\}\right)^2$
$x_4^u = v_2^u$	$x_4^o = v_2^o$
$\sigma_4^{uu} = \frac{-\left(v_2^U\right)^2 + \left(v_2^L\right)^2}{v_2^U - v_2^L}$	$\sigma_4^{uo} = \frac{-\left(v_2^U\right)^2 + \left(v_2^L\right)^2}{v_2^U - v_2^L}$
$\sigma_4^{ou} = -2\, v_2^u$	$\sigma_4^{oo} = -2\, v_2^o$
$\mathbf{s}_4^u = \begin{cases} \mathbf{0} & \text{if } x_4^u \leq x_4^{\min} \leq x_4^o, \\ \sigma_4^{uo}\mathbf{s}_2^o & \text{if } x_4^o < x_4^{\min}, \\ \sigma_4^{uo}\mathbf{s}_2^u & \text{otherwise,} \end{cases}$	$\mathbf{s}_4^o = \begin{cases} \mathbf{0} & \text{if } x_4^u \leq x_4^{\max} \leq x_4^o, \\ \sigma_4^{oo}\mathbf{s}_2^o & \text{if } x_4^o < x_4^{\max}, \\ \sigma_4^{ou}\mathbf{s}_2^u & \text{otherwise.} \end{cases}$
$v_5^L = v_3^L + v_4^L$	$v_5^U = v_3^U + v_4^U$
$v_5^u = v_3^u + v_4^u$	$v_5^o = v_3^o + v_4^o$
$\mathbf{s}_5^u = \mathbf{s}_3^u + \mathbf{s}_4^u$	$\mathbf{s}_5^o = \mathbf{s}_3^o + \mathbf{s}_4^o$
$a = v_1^L\, v_5^L$	$b = v_1^U\, v_5^U$
$c = v_1^U\, v_5^L$	$d = v_1^L\, v_5^U$
$v_6^L = \min\{a,b,c,d\}$	$v_6^U = \max\{a,b,c,d\}$
IF $v_5^L \geq 0$ THEN $\alpha_1 = v_5^L\, v_1^u,\ \mathbf{s}^{\alpha_1} = v_5^L\, \mathbf{s}_1^u$ ELSE $\alpha_1 = v_5^L\, v_1^o,\ \mathbf{s}^{\alpha_1} = v_5^L\, \mathbf{s}_1^o$	
IF $v_1^L \geq 0$ THEN $\alpha_2 = v_1^L\, v_5^u,\ \mathbf{s}^{\alpha_2} = v_1^L\, \mathbf{s}_5^u$ ELSE $\alpha_2 = v_1^L\, v_5^o,\ \mathbf{s}^{\alpha_2} = v_1^L\, \mathbf{s}_5^o$	
IF $v_5^U \geq 0$ THEN $\beta_1 = v_5^U\, v_1^u,\ \mathbf{s}^{\beta_1} = v_5^U\, \mathbf{s}_1^u$ ELSE $\beta_1 = v_5^U\, v_1^o,\ \mathbf{s}^{\beta_1} = v_5^U\, \mathbf{s}_1^o$	
IF $v_1^U \geq 0$ THEN $\beta_2 = v_1^U\, v_5^u,\ \mathbf{s}^{\beta_2} = v_1^U\, \mathbf{s}_5^u$ ELSE $\beta_2 = v_1^U\, v_5^o,\ \mathbf{s}^{\beta_2} = v_1^U\, \mathbf{s}_5^o$	
IF $v_5^L \geq 0$ THEN $\gamma_1 = v_5^L\, v_1^o,\ \mathbf{s}^{\gamma_1} = v_5^L\, \mathbf{s}_1^o$ ELSE $\gamma_1 = v_5^L\, v_1^u,\ \mathbf{s}^{\gamma_1} = v_5^L\, \mathbf{s}_1^u$	
IF $v_1^U \geq 0$ THEN $\gamma_2 = v_1^U\, v_5^o,\ \mathbf{s}^{\gamma_2} = v_1^U\, \mathbf{s}_5^o$ ELSE $\gamma_2 = v_1^U\, v_5^u,\ \mathbf{s}^{\gamma_2} = v_1^U\, \mathbf{s}_5^u$	
IF $v_5^U \geq 0$ THEN $\delta_1 = v_5^U\, v_1^o,\ \mathbf{s}^{\delta_1} = v_5^U\, \mathbf{s}_1^o$ ELSE $\delta_1 = v_5^U\, v_1^u,\ \mathbf{s}^{\delta_1} = v_5^U\, \mathbf{s}_1^u$	
IF $v_1^L \geq 0$ THEN $\delta_2 = v_1^L\, v_5^o,\ \mathbf{s}^{\delta_2} = v_1^L\, \mathbf{s}_5^u$ ELSE $\delta_2 = v_1^L\, v_5^u,\ \mathbf{s}^{\delta_2} = v_1^L\, \mathbf{s}_5^u$	
IF $\alpha_1 + \alpha_2 - a \geq \beta_1 + \beta_2 - b$ THEN $v_6^u = \alpha_1 + \alpha_2 - a,\ \mathbf{s}_6^u = \mathbf{s}^{\alpha_1} + \mathbf{s}^{\alpha_2}$ ELSE $v_6^u = \beta_1 + \beta_2 - b,\ \mathbf{s}_6^u = \mathbf{s}^{\beta_1} + \mathbf{s}^{\beta_2}$	
IF $\gamma_1 + \gamma_2 - c \leq \delta_1 + \delta_2 - d$ THEN $v_6^o = \gamma_1 + \gamma_2 - c,\ \mathbf{s}_6^o = \mathbf{s}^{\gamma_1} + \mathbf{s}^{\gamma_2}$ ELSE $v_6^o = \delta_1 + \delta_2 - d,\ \mathbf{s}_6^o = \mathbf{s}^{\delta_1} + \mathbf{s}^{\delta_2}$	

TABLE 4.1
continued.

```
$a = v_2^L\, v_6^L$                    $b = v_2^U\, v_6^U$
$c = v_2^U\, v_6^L$                    $d = v_2^L\, v_6^U$
$v_7^L = \min\{a, b, c, d\}$ $v_7^U = \max\{a, b, c, d\}$
IF $v_6^L \geq 0$ THEN $\alpha_1 = v_6^L\, v_2^u,\ \mathbf{s}^{\alpha_1} = v_6^L\, \mathbf{s}_2^u$ ELSE $\alpha_1 = v_6^L\, v_2^o,\ \mathbf{s}^{\alpha_1} = v_6^L\, \mathbf{s}_2^o$
IF $v_2^L \geq 0$ THEN $\alpha_2 = v_2^L\, v_6^u,\ \mathbf{s}^{\alpha_2} = v_2^L\, \mathbf{s}_6^u$ ELSE $\alpha_2 = v_2^L\, v_6^o,\ \mathbf{s}^{\alpha_1} = v_2^L\, \mathbf{s}_6^o$
IF $v_6^U \geq 0$ THEN $\beta_1 = v_6^U\, v_2^u,\ \mathbf{s}^{\beta_1} = v_6^U\, \mathbf{s}_2^u$ ELSE $\beta_1 = v_6^U\, v_2^o,\ \mathbf{s}^{\beta_1} = v_6^U\, \mathbf{s}_2^o$
IF $v_2^U \geq 0$ THEN $\beta_2 = v_2^U\, v_6^u,\ \mathbf{s}^{\beta_2} = v_2^U\, \mathbf{s}_6^u$ ELSE $\beta_2 = v_2^U\, v_6^o,\ \mathbf{s}^{\beta_2} = v_2^U\, \mathbf{s}_6^o$
IF $v_6^L \geq 0$ THEN $\gamma_1 = v_6^L\, v_2^o,\ \mathbf{s}^{\gamma_1} = v_6^L\, \mathbf{s}_2^o$ ELSE $\gamma_1 = v_6^L\, v_2^u,\ \mathbf{s}^{\gamma_1} = v_6^L\, \mathbf{s}_2^u$
IF $v_2^U \geq 0$ THEN $\gamma_2 = v_2^U\, v_6^o,\ \mathbf{s}^{\gamma_2} = v_2^U\, \mathbf{s}_6^o$ ELSE $\gamma_2 = v_2^U\, v_6^u,\ \mathbf{s}^{\gamma_2} = v_2^U\, \mathbf{s}_6^u$
IF $v_6^U \geq 0$ THEN $\delta_1 = v_6^U\, v_2^o,\ \mathbf{s}^{\delta_1} = v_6^U\, \mathbf{s}_2^o$ ELSE $\delta_1 = v_6^U\, v_2^u,\ \mathbf{s}^{\delta_1} = v_6^U\, \mathbf{s}_2^u$
IF $v_2^L \geq 0$ THEN $\delta_2 = v_2^L\, v_6^o,\ \mathbf{s}^{\delta_2} = v_2^L\, \mathbf{s}_6^u$ ELSE $\delta_2 = v_2^L\, v_6^u,\ \mathbf{s}^{\delta_2} = v_2^L\, \mathbf{s}_6^u$
IF $\alpha_1 + \alpha_2 - a \geq \beta_1 + \beta_2 - b$ THEN $v_7^u = \alpha_1 + \alpha_2 - a,\ \mathbf{s}_7^u = \mathbf{s}^{\alpha_1} + \mathbf{s}^{\alpha_2}$
    ELSE $v_7^u = \beta_1 + \beta_2 - b,\ \mathbf{s}_7^u = \mathbf{s}^{\beta_1} + \mathbf{s}^{\beta_2}$
IF $\gamma_1 + \gamma_2 - c \leq \delta_1 + \delta_2 - d$ THEN $v_7^o = \gamma_1 + \gamma_2 - c,\ \mathbf{s}_7^o = \mathbf{s}^{\gamma_1} + \mathbf{s}^{\gamma_2}$
    ELSE $v_7^o = \delta_1 + \delta_2 - d,\ \mathbf{s}_7^o = \mathbf{s}^{\delta_1} + \mathbf{s}^{\delta_2}$
```

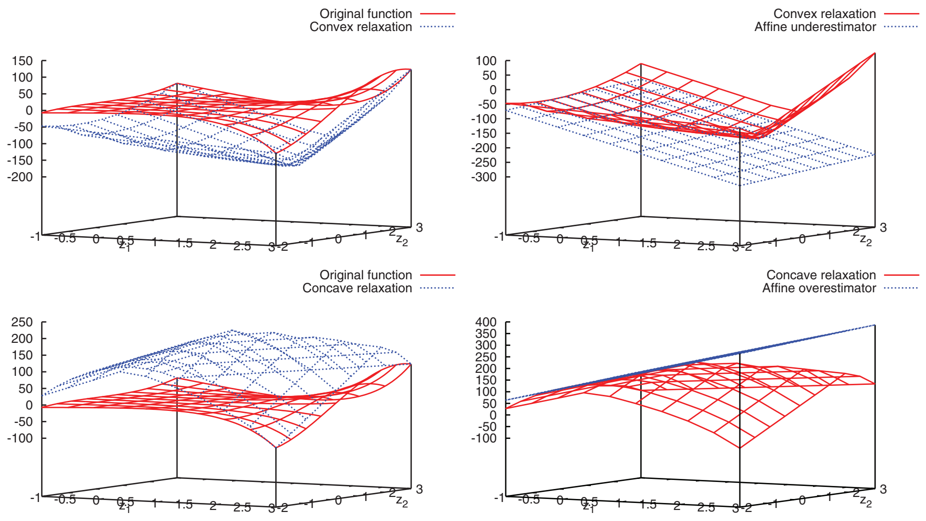


FIG. 4.3. *Graphical illustration of Example* 4.2.

Based on these relaxations and subgradients, an affine underestimator and an affine overestimator of g can be constructed by linearization (see subsection 3.2.1). One such affine underestimator and affine overestimator of g at $\bar{\mathbf{z}} = \mathbf{0}$ are shown in the upper and lower right plots of Figure 4.3. Observe, in particular, that neither the convex relaxation g^u nor the concave relaxation g^o are differentiable at $\bar{\mathbf{z}}$.

4.3. Implementation. From an implementation viewpoint, the evaluation of relaxations and subgradients can be automated in two ways: *operator overloading* and *source code transformation* [13]. In the source code transformation approach, a compiler is used to transform the source code for a function into another source code that includes statements for calculating the additional numbers (upper/lower bounds, convex/concave relaxations, and subgradients) interleaved with the original instructions. Source code transformation can be implemented for all programming languages, such as Fortran or C/C++. In the operator overloading approach, the implementer defines class objects for which the assignment, standard operators of arithmetic, and a collection of univariate intrinsic functions are overloaded via special language facilities to perform the desired calculation rules. The compiler is then responsible for applying the appropriate operations based on the runtime-type identification of the objects participating in expressions coded by the user so that no change in the original source code for the function is required. Overloading is, by nature, myopic in the sense that it does one operation at a time without any regard for the calculations occurring before and after. Normally, source code transformation can be smarter by performing some optimizations, such as reusing common subexpressions, but this improved efficiency comes at the expense of a much more sophisticated implementation. Because the primary motivation of this paper is to explore new algorithms and see whether they merit further investigation, the simpler-to-implement alternative of operator overloading approach is chosen. It is worth pointing out that for many small- to medium-size problems, particularly those where the relaxation calculation does not account for a high proportion of the runtime requirements, a simple overloading approach is perfectly adequate.

The implementation, libMC, is written in C++ and comes as an open source library [8]. In libMC, a new data type (class) for the program variables, named McCormick, is introduced that contains fields for storing the upper/lower bounds, convex/concave relaxations, and subgradients of the relaxations:

```
Class McCormick{
  double  _x;     // current value
  double  _l;     // lower bound
  double  _u;     // upper bound
  double  _cv;    // convex relaxation
  double  _cc;    // concave relaxation
  int     _np;    // number of independent variables
  double* _dcvdp; // subgradient of convex relaxation
  double* _dccdp; // subgradient of concave relaxation
  ...
}
```

`libMC` calculates the values of these fields simultaneously with the function values, in accordance with the operations in the program. The operators for addition, subtraction, multiplication, and division are supported, as well as the standard exponential, logarithmic, power, square root, and absolute value intrinsic functions. Note, however, that monomials of odd degree, such as y^{2n+1}, are treated as bilinear terms of the form $y \times y^{2n}$. Note also that the function $y \mapsto y \ln y$, which is convex on $[0, +\infty)$ and differentiable on $(0, +\infty)$, is supported as an intrinsic function. As just one example, the operator * for the multiplication of a McCormick variable with a scalar is implemented as follows:

```
McCormick operator*( const double scal, const McCormick &MC ){
  McCormick MCres;
  if ( scal >= 0 ){
    // lower/upper bounds
    MCres._l = scal * MC._l;
    MCres._u = scal * MC._u;
    // convex/concave relaxations
    MCres._cv = scal * MC._cv;
    MCres._cc = scal * MC._cc;
    // convex/concave relaxation subgradients
    for( int ip=0; ip<MCres._np; ip++ ){
      MCres._dcvdp[ip] = scal * MC._dcvdp[ip];
      MCres._dccdp[ip] = scal * MC._dccdp[ip];
    }
  }
  else{
    // lower/upper bounds
    MCres._l = scal * MC._u;
    MCres._u = scal * MC._l;
    // convex/concave relaxations
    MCres._cv = scal * MC._cc;
    MCres._cc = scal * MC._cv;
    // convex/concave relaxation subgradients
    for( int ip=0; ip<MCres._np; ip++ ){
      MCres._dcvdp[ip] = scal * MC._dccdp[ip];
      MCres._dccdp[ip] = scal * MC._dcvdp[ip];
    }
  }
  return MCres;
}
```

An example illustrating the use of libMC is presented subsequently.

Example 4.3 (use of libMC). Consider the same function g as in Examples 4.1 and 4.2, for which values of lower/upper bounds, convex/concave relaxations, and subgradients are to be evaluated at the point $\bar{\mathbf{z}} = \mathbf{0}$ for $\mathbf{z} \in Z$. First, the number of independent variables is specified, and each independent variable is initialized:

```
McCormick::np(2);
McCormick Z1( -1., 3., 0., 0 );
McCormick Z2( -2., 3., 0., 1 );
```

Essentially, the first line states that the function has two independent variables. The independent variables are indexed in the range $\{0, 1, \ldots, np-1\}$ for ease of access to the elements of the subgradients of the function g. The second line states that Z1 is a variable of class McCormick, with range $[-1, 3]$, current value 0, and index 0. Similarly, the third line defines the variable Z2 to have range $[-2, 3]$, current value 0, and index 1.

Next, the bounds, relaxations, and subgradients for g at $\bar{\mathbf{z}}$ are simply calculated as

```
McCormick G = (exp(Z1)-pow(Z2,2))*Z1*Z2;
```

The resulting values for the lower and upper bounds are retrieved as

```
double GL = G.l();
double GU = G.u();
```

convex and concave relaxations as

```
double Gu = G.cv();
double Go = G.cc();
```

and a subgradient of each of these relaxations as

```
double* Su = G.dcvdp();
double* So = G.dccdp();
```

The current implementation of libMC allows only the propagation of a single subgradient for the convex relaxation and of another subgradient for the concave relaxation. To illustrate this limitation, consider the bilinear term $g(\mathbf{z}) = z_1 z_2$ on $Z = [\mathbf{z}^L, \mathbf{z}^U]$. A convex relaxation of g on Z is $g^u(\mathbf{z}) = \max\{z_1 z_2^L + z_1^L z_2 - z_1^L z_2^L, z_1 z_2^U + z_1^U z_2 - z_1^U z_2^U\}$, and the corresponding subdifferential of g^u at a point $\bar{\mathbf{z}}$ such that $\bar{z}_1 = \bar{z}_2$ is $\partial g^u(\bar{\mathbf{z}}) = \text{conv}\{\mathbf{s}^1, \mathbf{s}^2\}$, with $\mathbf{s}^1 = (z_2^L, z_1^L)$ and $\mathbf{s}^2 = (z_2^U, z_1^U)$, where conv denotes the convex hull. In this example, libMC returns either $\mathbf{s}^1$ or $\mathbf{s}^2$ as a subgradient, depending on round-off errors.

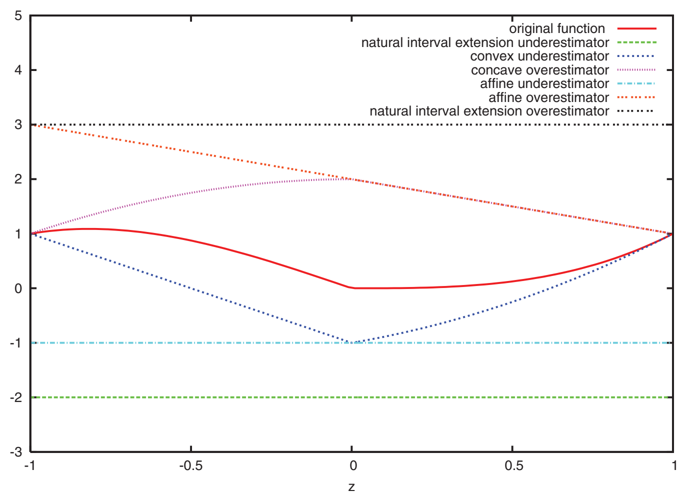


FIG. 4.4. *Graphical illustration of Example* 4.4.

A possible extension of libMC would be to propagate multiple subgradients at a point whenever nonsmoothness is detected in the relaxations. In particular, Theorem 3.2 provides three possible subgradients at a point where the convex or concave relaxation of a univariate function term is nonsmooth; likewise, Theorem 3.3 provides two possible subgradients at a point where the convex or concave relaxation of a binary product term is nonsmooth. Then, noting that any convex combination of subgradients at a point of a relaxation yields another valid subgradient at that point, the user would be given the ability to choose among multiple possible subgradients. Due to round-off errors, however, there is little chance that a point of nonsmoothness between two or more smooth parts is hit at a generic argument $\mathbf{z}$ whose components have been perturbed away from special values such as 0, 1, etc. For this reason, even though the theory allows to propagate several subgradients, only a subset of the entire subdifferential can be obtained in most practical problems.

4.4. Illustrative example. The following illustrative example is based on `libMC` for the bound, relaxation, and subgradient calculations.

Example 4.4 (univariate function). Let $Z = [-1, 1]$, $\bar{z} = 0$, and $g(z) = |z| + z^3 - z$, with the global minimum 0 attained at $z = 0$ and the global maximum 1.0886 attained at $z = -0.8165$.

Natural interval extensions of this function give a lower bound of -2 and an upper bound of 3. The McCormick relaxation gives a lower bound of -1 attained at $z = 0$ and an upper bound of 2 also attained at $z = 0$. The implemented code calculates (0) as a subgradient to g^u at $\bar{z}$ and (-1) as a subgradient to g^o at $\bar{z}$. The (constant) affine underestimator described in section 3.2.1 gives a lower bound of -1 and an upper bound of 3 attained at $z = -1$.

As is seen in Figure 4.4, the affine underestimator is a constant, which is tighter than the lower bound provided by the natural interval extensions for all $z \in Z$. The affine overestimator is tighter than the upper bound provided by the natural interval extensions for all $z > -1$ and equally tight for $z = -1$. A different choice of subgradient would have given a tighter affine overestimator; for instance, (0) would have given 2 as the overestimator.

5. Relaxations of algorithms. So far in all examples, the functions were explicitly known. However, the theoretical results developed here are directly applicable also to functions calculated by algorithms. As discussed in the introduction, this propagation of McCormick convex/concave relaxations and their subgradients allows the relaxation and global optimization of certain algorithms. The class of algorithms considered here has two main restrictions. First, it is assumed that the algorithms take a fixed number of iterations, known a priori. For instance, direct solvers for linear equation systems satisfy this assumption, while the Newton–Raphson method for the solution of nonlinear equation systems does not. The second restriction imposed is that the algorithms consist of only addition/subtraction, multiplication/division, and intrinsic functions (with known relaxations and enclosures). For instance, algorithms with conditional statements (e.g., `IF` $f < 0$ `THEN` $x = xL$ `ELSE` $x = xU$) are not yet considered. An example of an algorithm that requires such conditional statements is the bisection method for the solution of a nonlinear equation.

Building upon the theoretical framework described, `libMC` can be readily applied to the relaxation of algorithms. To do so, the algorithm is encoded using the `libMC` data structures (as opposed to the native data structures of common programming languages), and `libMC` evaluates the convex/concave relaxations as well as a subgradient.

5.1. Numerical examples. In this section the affine relaxation of two algorithms with a fixed number of iterations is considered, namely, the solution of the one-dimensional heat equation via finite differences and the solution of a nonlinear ODE system via the explicit-Euler method. In both cases a nonconvex parameter estimation problem is considered. Recall that efficient solvers for convex NLPs involving nonsmooth functions have been recently developed, e.g., [20]. The results and implementation presented here would facilitate global optimization using such solvers in sophisticated global optimization algorithms such as branch-and-reduce. However, this would raise several implementation issues. Therefore, the simpler case of affine relaxations (see section 3.2.1) using function evaluations is, instead, considered in the following using a simple branch-and-bound procedure.

The parameter estimations considered can be summarized as

$$\begin{aligned}&\min_{\mathbf{p}} \sum_{i=1}^{n_m} \left(y(\mathbf{p}, q_i) - y^m(q_i)\right)^2\\ &\text{s.t. } \mathbf{p} \in \left[\mathbf{p}^L, \mathbf{p}^U\right],\end{aligned} \tag{5.1}$$

where the *parameters* $\mathbf{p} \in \mathbb{R}^{n_p}$ are only subject to box constraints. The *independent variable* q is time t or space x; q_i refers to specific values of the independent variable at which the *measurements* $y^m(q_i)$ are obtained. The *output variable* $y(\mathbf{p}, q_i)$ is evaluated as a function of the parameters through an algorithm with a fixed number of iterations. The dependence on the parameters is nonconvex and not known explicitly. The goal of the parameter estimation is to find a parameter point that matches the measurements in an optimal fashion. Here, for simplicity, the least-squares error objective function is used as a metric of fit. Application to other objectives poses no theoretical or algorithmic difficulties.

In both cases the *state variables* are given by the solution of an ODE system. The state variables are discretized and calculated by a simple algorithm (i.e., an algorithm with a fixed number of iterations). The output variable is a weighted sum of the state variables. The procedures to calculate $y(\mathbf{p}, q_i)$ are encoded in `libMC`, which evaluates the convex/concave relaxations as well as a subgradient at a point $\mathbf{p}$. The affine lower bound described in section 3.2.1 gives a lower bound to the optimal objective value of (5.1). As a proof of concept, the lower bound is converged by a simple branch-and-bound procedure in the parameters $\mathbf{p}$ with a best-bound node selection heuristic and with bisection on the parameter range with the largest diameter. The midpoint of each node is used for the linearization. Significant CPU savings could be achieved with a more sophisticated branch-and-reduce algorithm. Also savings could be achieved if a priori information on the state variable bounds were used, as is done in [30].

The commercial code BARON version 8.1.5 [28] available through GAMS version 22.8 [5] is used as a benchmark for the performance of the proposed algorithmic relaxations. The results are reproduced with BARON version 7.5 through GAMS version 22.2 and BARON version 8.1.1 through GAMS version 22.6, and no significant differences between the three versions are found. BARON requires explicit functions, and therefore, the discretized state variables are encoded as optimization variables along with the equality constraints from the discretization of the ODEs relating them. BARON implements a sophisticated branch-and-reduce algorithm based on the linearization of a smooth convex relaxation using auxiliary variables [34]. By introducing these auxiliary variables, the convex/concave relaxations can become tighter [34]. The termination criteria of BARON are set to $\texttt{OPTCA} = 10^{-6}$ and $\texttt{OPTCR} = 10^{-6}$. This ensures that the lower bound displayed is truly a lower bound. The resource limit is set to 30 CPU minutes ($\texttt{RESLIM} = 1800$). Since BARON is largely a black-box solver and the two methods are very different, the comparison is done in terms of the CPU requirement (on a 64-bit Intel core2 DUO processor ULV7700 at 1.33GHz running Linux 2.6.25). An obvious advantage of the lower bounding scheme proposed here is the significantly smaller number of optimization variables. As shown by the results following, this advantage can result in drastically faster convergence of the lower bound. BARON is a state-of-the-art deterministic global NLP solver, and therefore, the comparison shows the promise of the proposed method. It should, however, be noted that the problems have a special structure exploited by the proposed method; no claim is made that the proposed affine relaxations outperform other approaches such as the one implemented in BARON in general.

In addition, the proposed method is compared with BARON, using optimization variables and equality constraints for the state variables, but without branching on the state variables. This can be interpreted as relaxation of algorithms with auxiliary variables and to the best knowledge of the authors, it has not been proposed in the literature. A validity proof of this approach follows similar arguments as the proposed method and is omitted for brevity. The disadvantage compared to the proposed method is that the size of the lower and upper bounding problems is significantly larger. An advantage is potentially tighter relaxations. This *selective branching* significantly reduces the computational requirements. Nevertheless, the proposed method is significantly faster, in particular for the heat equation example. The selective branching is implemented using the attribute `.prior` with a value of 0 for the variables for which no branching should be performed; this attribute is specified in the BARON options file.

The case studies have an imperfect match between model and measurement. In both case studies finding a global minimum is relatively simple, e.g., through a multistart method, and the challenge for global optimization methods is to converge the lower bound. Establishing that the lower bound is higher than the statistical test for the fit proves that the mismatch is due to the model and not to a suboptimal parameter fit [31]. This is a major advantage of using global optimization techniques in parameter estimation problems. Note that, by construction, for both BARON and the McCormick relaxations, the lower bounds are always nonnegative. As a consequence, parameter estimation problems with a perfect match between model and experiment are not interesting regarding the convergence of the lower bound.

5.1.1. Heat equation. The one-dimensional heat equation with an affine heat source term

$$\frac{d^2T}{dx^2} = -\frac{q^0(x) + q^1(x)T}{p} \tag{5.2}$$

is linear in the state variable T. The thermal conductivity p is taken as the (single) unknown parameter in the range $p \in [0.01, 10]$. Note that dividing by the thermal conductivity results in better numerical behavior. To determine the dependence on the position $x \in [0, 1]$, two boundary conditions are needed. Here, the simplest case of known temperature at the boundary points is taken:

$$T(x = 0) = 500 \quad \text{and} \quad T(x = 1) = 600.$$

The temperature-dependent heat source term $q^1(x)$ is taken as constant, i.e., $q^1(x) = -1$, while the temperature-independent heat source term is taken as

$$q^0(x) = \begin{cases} 35,000 & \text{if } x \in [0.5, 0.6], \\ -5,000 & \text{otherwise.} \end{cases}$$

This boundary value problem can be approximated by a discretization of the independent variable. Here, an equidistant discretization into $n-1$ intervals is considered, with $n = 101$. The state variable is discretized on this mesh as T_i, $i = 1, \dots, n$ and so is the x-dependent heat-source term q^1. The central-difference formula is used for the second derivative:

$$\frac{d^2T_i}{dx^2} \approx \frac{T_{i-1} - 2T_i + T_{i+1}}{\Delta x^2}, \qquad i \in \{2, 3, \dots, n-1\},$$

where $\Delta x = 1/(n-1)$.

By introducing auxiliary variables for the discretized state variable, the approximated problem can be encoded as a regular NLP. BARON requires all variables and functions to be bounded, and therefore, the temperature is restricted to $T_i \in [0, 2000]$. With this addition, the NLP satisfies all assumptions of BARON and can (at least in principle) be solved for an arbitrary n. The resulting NLP has $n-1$ variables. Note also that BARON has sophisticated rules to tighten the range of the variables, including constraint propagation.

The alternative considered here is to exploit the linearity of (5.2) in the variables T_i:

$$\begin{bmatrix} 1 & & & & & & \\ 1 & \left(-2 - \frac{q_2^1}{p}\Delta x^2\right) & 1 & & & & \\ & 1 & \left(-2 - \frac{q_3^1}{p}\Delta x^2\right) & 1 & & & \\ & & \dots & \dots & \dots & & \\ & & & \dots & \dots & \dots & \\ & & & & 1 & \left(-2 - \frac{q_{n-1}^1}{p}\Delta x^2\right) & 1 \\ & & & & & & 1 \end{bmatrix} \begin{bmatrix} T_1 \\ T_2 \\ T_3 \\ \dots \\ \dots \\ T_{n-1} \\ T_n \end{bmatrix} = \begin{bmatrix} 500 \\ -\frac{q^0}{p}\Delta x^2 \\ -\frac{q^0}{p}\Delta x^2 \\ \dots \\ \dots \\ -\frac{q^0}{p}\Delta x^2 \\ 600 \end{bmatrix}.$$

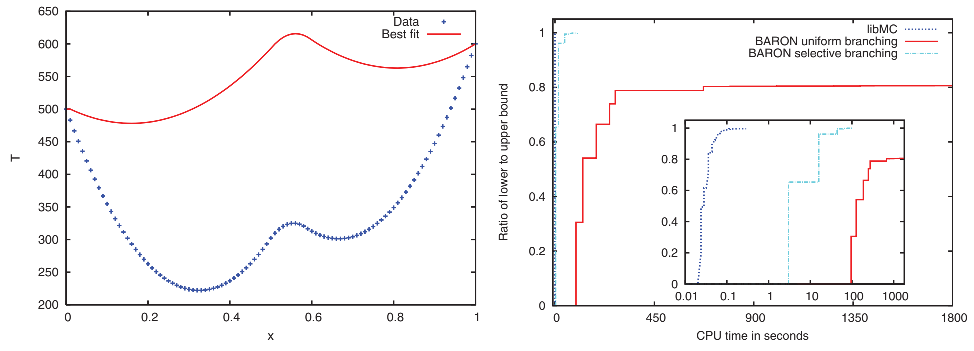


FIG. 5.1. *Graphical illustration of heat equation example: data and best fit (left plot) and convergence of lower bound (right plot). Note that in the right plot, both a semilogarithmic and a linear plot are shown.*

For a given value of p, this linear equation system can be solved by forward elimination and back-substitution in $O(n)$ operations. The storage requirement is less than $4n$ intermediate variables. The solution vector $\mathbf{T}$ can then be used to calculate the objective function in (5.1). This simple algorithm can be coded in `libMC` which, in addition to the values of T_i, calculates their convex/concave relaxations on $[\mathbf{p}^L, \mathbf{p}^U]$. This allows the global optimization of the algorithm. This optimization problem has a single variable. The bounds for the state variables are not used here.

The measurements $\mathbf{T}^m$ are generated with a different heat source term to simulate the case where the model is not correct. The (x_i, T_i^m) pairs used as data are given in [8]: http://yoric.mit.edu/libMC/libmcheatexdata.txt. Finding a global minimum to the parameter estimation is relatively simple. However, the fit is not satisfactory, as can be seen in Figure 5.1. In a real-world application this would raise the question if the fit is suboptimal or if the model cannot represent the data. A certificate of optimality given by a global optimization algorithm can exclude the former case.

Figure 5.1 shows the ratio of lower bound to upper bound versus CPU time in seconds for the three methods. Clearly, the relaxation of the algorithm is computationally far superior over introducing optimization variables for the discretized state. In the proposed method the affine relaxation converges to the upper bound within 99% in just 0.10 seconds and to 99.5% in just 0.12 seconds. In contrast BARON requires 95 seconds to reach 30.5% convergence and 87 seconds to reach 66% convergence. After 1800 seconds BARON is still at 80% convergence. It should again be noted that BARON is the state-of-the-art in general-purpose deterministic global optimization algorithms, and therefore, this comparison shows the potential of algorithmic relaxation. The proposed method is significantly faster than selective branching, but the difference is much smaller. When branching is not performed on the state variables, BARON requires 3 seconds to reach 65% convergence and 16 seconds to reach 96% convergence.

It should be noted that both the proposed method and BARON suffer from discretization error. To estimate the discretization error introduced, the heat conductivity p is fixed to its optimal value and the temperature predictions $T(x)$ are compared for $n = 101$ and $n = 1001$. The difference between the coarser and finer mesh are in the order of 10%, which is much smaller than the error between the measurements and model (more than 100%). Therefore, the discretization is deemed sufficiently fine, especially in view of the high CPU requirement by BARON.

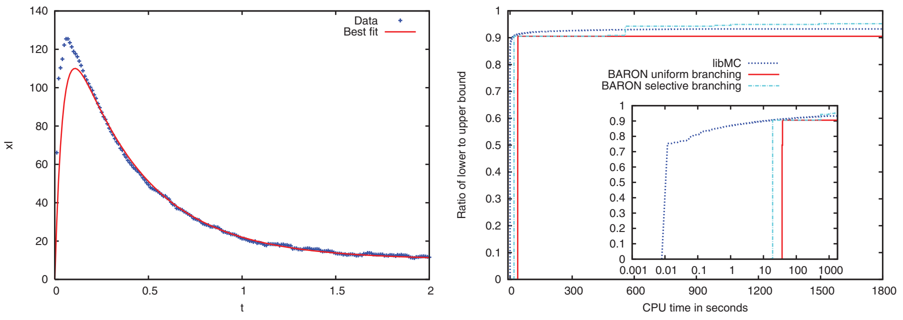


FIG. 5.2. *Graphical illustration of kinetic mechanism example: data and best fit (left plot) and convergence of lower bound (right plot). Note that in the right plot, both a semilogarithmic and a linear plot are shown.*

5.1.2. Kinetic mechanism. Here, an example from chemical kinetics is considered based on [37, 31, 29]. Mathematically, it is a nonlinear ODE system:

$$\begin{aligned}
&\frac{dx_A}{dt} = k_1\, x_Z\, x_Y - x_{\mathrm{O}_2}\,(k_{2f} + k_{3f})\, x_A + \frac{k_{2f}}{K_2}\, x_D + \frac{k_{3f}}{K_3}\, x_B - k_5\, x_A^2,\\
&\frac{dx_Z}{dt} = -k_1\, x_Z\, x_Y, \qquad \frac{dx_Y}{dt} = -k_{1s}\, x_Z\, x_Y,\\
&\frac{dx_D}{dt} = k_{2f}\, x_A\, x_{\mathrm{O}_2} - \frac{k_{2f}}{K_2}\, x_D, \qquad \frac{dx_B}{dt} = k_{3f}\, x_{\mathrm{O}_2}\, x_A - \left(\frac{k_{3f}}{K_3} + k_4\right) x_B,\\
&x_A(t=0) = 0,\ x_B(t=0) = 0,\ x_D(t=0) = 0,\ x_Y(t=0) = 0.4,\ x_Z(t=0) = 140,
\end{aligned}$$

with the constants $T = 273$, $K_2 = 46\, e^{\frac{6500}{T} - 18}$, $K_3 = 2K_2$, $k_1 = 53$, $k_{1s} = k_1 \times 10^{-6}$, $k_5 = 0.0012$, and $x_{\mathrm{O}_2} = 0.002$. The unknown parameters are $k_{2f} \in [10, 1200]$, $k_{3f} \in [10, 1200]$, and $k_4 \in [0.001, 40]$. After scaling, the measurements are given in terms of $x_I = x_A + \frac{2}{21}\, x_B + \frac{2}{21}\, x_D$ (without introducing an additional variable). The $(t_i, x_{I,i}^m)$ pairs used as data are given in [8]: http://yoric.mit.edu/libMC/libmckinexdata.txt. Finding a global minimum to the parameter estimation is relatively simple. The fit is reasonably good, but for initial times, there is a significant discrepancy, as can be seen in Figure 5.2, raising the question if the fit is suboptimal or if the model cannot represent the data. A certificate of optimality given by a global optimization algorithm can exclude the former case.

The explicit-Euler method can be used to approximate the solution of this initial-value problem. The time is discretized in n equidistant intervals, and the state variables are discretized on this mesh. Here, $n = 200$ is used.

By introducing auxiliary variables for the discretized state variable, the approximated problem can be encoded as a regular NLP. BARON requires all variables and functions to be bounded, and therefore, the states are restricted to $x_i \in [0, 140]$, $i \neq Y$, and $x_Y \in [0, 0.4]$. With this addition, the NLP satisfies all assumptions of BARON and can (at least in principle) be solved for an arbitrary n. The resulting NLP has $5n + 3$ variables. Recall that BARON has sophisticated rules to tighten the range of the variables.

The alternative considered here is to encode the explicit-Euler method in `libMC`, which calculates the convex/concave relaxations of the state variables. This allows the global optimization of the algorithm. This optimization problem has only three variables (the unknown parameters). The bounds for the state variables are not used here.

Recently, sophisticated global optimization algorithms for initial-value problems have been proposed; see [9] for an overview. At this point it is, therefore, interesting to compare these algorithms conceptually with the proposed relaxation of algorithms. Both methods rely on discretization of the independent variable and on McCormick relaxations. However, in the existing algorithms, relaxation is performed before the discretization (done by the integrator), whereas here the relaxation is performed after the discretization. Another difference is that here state bounds (via natural interval extensions) and relaxations are obtained simultaneously, whereas in the existing algorithms they are treated independently. An interesting observation is that the relaxation of algorithms provides a very simple alternative to the elaborate theory of [9].

Figure 5.2 shows the ratio of lower bound to upper bound versus CPU time in seconds for the two methods. Again, the relaxation of the algorithm is computationally superior over introducing optimization variables for the discretized state. In the proposed method the affine relaxation converges to the upper bound within 75% in just 0.012 seconds, to 80% in just 0.07 seconds, and to 90% in 11 seconds. In contrast BARON requires 37 seconds to furnish a positive lower bound (equal to 75% of the upper bound) and 38 seconds to reach 90.5%. Moreover, the proposed method shows continued (albeit slow) improvement, whereas BARON shows no improvement between 44 and 1800 seconds. It should again be noted that BARON is the state-of-the-art in general-purpose deterministic global optimization algorithms, and therefore, this comparison shows the potential of algorithmic relaxation. The proposed method is initially significantly faster than selective branching. When branching is not performed on the state variables, BARON requires 19 seconds to reach 90% convergence. After 557 seconds, BARON with selective branching reaches 95% convergence, which is better than the proposed method. This is most likely due to the superior branching strategy and range reduction implemented in BARON. Note also that the algorithm used in [31] appears to be more efficient; this is expected, since it is a specialized algorithm, uses a priori knowledge on the bounds, range reduction, and scaling of the optimization variables.

It should be noted that both the proposed method and BARON suffer from discretization error. To estimate the discretization error introduced, the kinetic constants k_{2f}, k_{3f}, and k_4 are fixed to their optimal values, and $x_i(t)$ is compared with a backward differentiation formula (BDF) method [38]. The difference between these two integration methods is in the order of 10% for initial times and then falls very quickly to approximately 1%. This difference is much smaller than the error between the measurements and model (more than 100% for initial times). Therefore, the discretization is deemed sufficiently fine, especially in view of the high CPU requirement.

6. Conclusions and future work.

Theory, application, and implementation for the computation of subgradients for McCormick relaxations were presented. This systematic subgradient propagation allows the affine over- and underestimation of factorable functions without the introduction of auxiliary variables. One potential application is the calculation of very cheap lower bounds for box-constrained problems; these bounds are often tighter than the bounds calculated by interval extensions with a similar computational requirement. Moreover, the affine relaxations are quite tight in the vicinity of the linearization points.

A limitation of the current implementation is that it cannot take advantage of the tightest possible affine relaxation in the case of nonunique subgradients. In general, it would be desirable to propagate the entire subdifferential or an easily characterized subset of it, as opposed to simply an element. This would facilitate a choice of subgradients. Furthermore, it would be interesting to consider the option of code generation. Another possible extension of libMC would be to propagate the subgradients of convex/concave relaxations via the reverse mode of AD (see, e.g., [13] for a general introduction on the reverse mode of AD). In this approach, the calculation of subgradients would be split into two steps: a forward sweep in which each intermediate values along with the corresponding lower/upper bounds and convex/concave relaxations are computed, similar to the approach described in subsection 4.2; followed by a reverse sweep in which the subgradients of the convex/concave relaxations are computed. This approach would be particularly efficient for problems having many independent variables and relatively few functions, e.g., in large-scale optimization problems.

Additional future work of interest is the application of the subgradient computation presented to global optimization of NLPs with a branch-and-bound code; instead of performing linearizations, it might be advantageous to directly solve the nonsmooth nonlinear relaxation with nonsmooth NLP solvers, such as bundle methods, e.g., [20, 15]; the proposed results provide the basis for the calculation of the subgradients required by these solvers. Additional problems of interest are semi-infinite programs and programs with differential equations embedded.

The proposed methodology sets the foundations for automatic convex relaxation of any algorithm, enabling the global optimization of algorithms. This was demonstrated on two simple algorithms with a fixed number of iterations. The computational requirement was drastically smaller than the alternative of introducing auxiliary variables. The computational procedure described here can presumably be improved significantly by using range reduction and employing nonlinear relaxations via nonsmooth NLP solvers. The demonstrated potential of algorithmic relaxation motivates future work in the relaxation of more general algorithms. In particular, it would be interesting to consider algorithms with a problem-dependent number of iterations, not known a priori. Such algorithms of interest include numerical integrators with error control and nonlinear equation solvers. The relaxation of such algorithms can potentially lead to global optimization of problems for which currently no algorithm exists, e.g., dynamic optimization with differential-algebraic equation systems embedded. There are several open challenges, including which algorithms can be sensibly relaxed and how the convergence of the relaxation relates to the convergence of the algorithm. It would be interesting to do a numerical and theoretical comparison of the proposed relaxations with the relaxations obtained by the use of auxiliary variables for the global optimization of general NLPs. No general claims can be made at this point, but the proposed relaxations are expected to outperform existing methods for a subclass of problems, e.g., those with many intermediate variables and few degrees of freedom.

Acknowledgments. We would like to thank Mehmet Yunt, Panayiotis Lemonidis, and Ajay Selot for fruitful discussions. We are grateful to Cha Kun Lee for the branch-and-bound code we derivatized. Finally, we would like to thank Nikolaos V. Sahinidis for pointing out the possibility to use selective branching in BARON.

REFERENCES

[1] C. S. Adjiman, S. Dallwig, C. A. Floudas, and A. Neumaier, *A global optimization method, αBB, for general twice-differentiable constrained NLPs* - I. *Theoretical advances*, Comput. Chem. Eng., 22 (1998), pp. 1137–1158.
[2] C. S. Adjiman and C. A. Floudas, *Rigorous convex underestimators for general twice-differentiable problems*, J. Global Optim., 9 (1996), pp. 23–40.

[3] G. Alefeld and G. Mayer, *Interval analysis: Theory and applications*, J. Comput. Appl. Math., 121 (2000), pp. 421–464.

[4] D. P. Bertsekas, *Nonlinear Programming*, 2nd ed., Athena Scientific, Belmont, MA, 1999.

[5] A. Brooke, D. Kendrick, and A. Meeraus, *GAMS: A User's Guide*, The Scientific Press, Redwood City, CA, 1988.

[6] R. P. Byrne and I. D. L. Bogle, *Global optimisation of constrained non-convex programs using reformulation and interval analysis*, Comput. Chem. Eng., 23 (1999), pp. 1341–1350.

[7] R. P. Byrne and I. D. L. Bogle, *Global optimization of modular process flowsheets*, Ind. Eng. Chem. Res., 39 (2000), pp. 4296–4301.

[8] B. Chachuat, `libMC`*: A Numeric Library for McCormick Relaxation of Factorable Functions*, documentation and source code available at: http://yoric.mit.edu/libMC/.

[9] B. Chachuat, A. B. Singer, and P. I. Barton, *Global methods for dynamic optimization and mixed-integer dynamic optimization*, Ind. Eng. Chem. Res., 45 (2006), pp. 8373–8392.

[10] J. E. Falk and R. M. Soland, *An algorithm for separable nonconvex programming problems*, Manag. Sci., 15 (1969), pp. 550–569.

[11] C. A. Floudas and O. Stein, *The adaptive convexification algorithm: A feasible point method for semi-infinite programming*, SIAM J. Optim., 18 (2007), pp. 1187–1208.

[12] E. P. Gatzke, J. E. Tolsma, and P. I. Barton, *Construction of convex function relaxations using automated code generation techniques*, Optim. Eng., 3 (2002), pp. 305–326.

[13] A. Griewank, *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*, Frontiers Appl. Math., SIAM, Philadelphia, 2000.

[14] J.-B. Hiriart-Urruty and C. Lemaréchal, *Convex Analysis and Minimization Algorithms* I. *Fundamentals*, Springer-Verlag, Berlin, 1993.

[15] J.-B. Hiriart-Urruty and C. Lemaréchal, *Convex Analysis and Minimization Algorithms* II. *Advanced Theory and Bundle Methods*, Springer-Verlag, Berlin, 1993.

[16] J.-B. Hiriart-Urruty and C. Lemaréchal, *Fundamentals of Convex Analysis*, Grundlehren Text Editions, Springer-Verlag, Berlin, 2001.

[17] R. Horst, *A general class of branch-and-bound methods in global optimization with some new approaches for concave minimization*, J. Optim. Theory Appl., 51 (1986), pp. 271–291.

[18] R. Horst and H. Tuy, *Global Optimization: Deterministic Approaches*, 3rd ed., Springer-Verlag, Berlin, 1996.

[19] P. Kesavan and P. I. Barton, *Generalized branch-and-cut framework for mixed-integer nonlinear optimization problems*, Comput. Chem. Eng., 24 (2000), pp. 1361–1366.

[20] M. M. Mäkelä and P. Neittaanmäki, *Nonsmooth Optimization: Analysis and Algorithms with Applications to Optimal Control*, World Scientific, Singapore, 1992.

[21] G. P. McCormick, *Computability of global solutions to factorable nonconvex programs: Part* I. *Convex underestimating problems*, Math. Program., 10 (1976), pp. 147–175.

[22] G. P. McCormick, *Nonlinear Programming: Theory, Algorithms and Applications*, John Wiley and Sons, New York, 1983.

[23] A. Mitsos, P. Lemonidis, C. K. Lee, and P. I. Barton, *Relaxation-based bounds for semi-infinite programs*, SIAM J. Optim., 19 (2008), pp. 77–113.

[24] R. Moore, *Methods and Applications of Interval Analysis*, SIAM, Philadelphia, 1979.

[25] A. Neumaier, *Complete search in continuous global optimization and constraint satisfaction*, Acta Numer., 13 (2004), pp. 271–369.

[26] R. T. Rockafellar, *Convex Analysis*, Princeton Mathematical Series 28, Princeton University Press, Princeton, NJ, 1970.

[27] H. S. Ryoo and N. V. Sahinidis, *A branch-and-reduce approach to global optimization*, J. Global Optim., 8 (1996), pp. 107–138.

[28] N. Sahinidis and M. Tawarmalani, *BARON*. http://www.gams.com/solvers/baron.pdf.

[29] A. B. Singer, *Global Dynamic Optimization*, Ph.D. Thesis, Massachusetts Institute of Technology, Cambridge, MA, 2004, http://yoric.mit.edu/download/Reports/SingerThesis.pdf.

[30] A. B. Singer and P. I. Barton, *Bounding the solutions of parameter dependent nonlinear ordinary differential equations*, SIAM J. Sci. Comput., 27 (2006), pp. 2167–2182.

[31] A. B. Singer, J. W. Taylor, P. I. Barton, and W. H. Green, *Global dynamic optimization for parameter estimation in chemical kinetics*, J. Phys. Chem. A, 110 (2006), pp. 971–976.

[32] E. M. B. Smith and C. C. Pantelides, *Global optimization of general process models*, in Global Optimization in Engineering Design, I. E. Grossmann, ed., Kluwer Academic Publishers, Norwell, MA, 1996, pp. 355–386.

[33] E. M. B. Smith and C. C. Pantelides, *A symbolic reformulation/spatial branch-and-bound algorithm for the global optimisation of nonconvex MINLPs*, Comput. Chem. Eng., 23 (1999), pp. 457–478.

[34] M. Tawarmalani and N. V. Sahinidis, *Convexification and Global Optimization in Continuous and Mixed-Integer Nonlinear Programming*, Nonconvex Optim. Appl., Kluwer Academic Publishers, Norwell, MA, 2002.

[35] M. Tawarmalani and N. V. Sahinidis, *Global optimization of mixed-integer nonlinear programs: A theoretical and computational study*, Math. Program., 99 (2004), pp. 563–591.

[36] M. Tawarmalani and N. V. Sahinidis, *A polyhedral branch-and-cut approach to global optimization*, Math. Program., 103 (2005), pp. 225–249.

[37] J. W. Taylor, G. Ehlker, H. H. Carstensen, L. Ruslen, R. W. Field, and W. H. Green, *Direct measurement of the fast, reversible addition of oxygen to cyclohexadienyl radicals in nonpolar solvents*, J. Phys. Chem. A, 108 (2004), pp. 7193–7203.

[38] J. Tolsma and P. I. Barton, *DAEPACK: An open modeling environment for legacy models*, Ind. Eng. Chem. Res., 39 (2000), pp. 1826–1839.

[39] X. J. Wang, *Global and Local Optimization Using Linear Bounding Functions*, Ph.D. thesis, University of California, Davis, CA, 1995.

[40] X. J. Wang and T. S. Chang, *An improved univariate global optimization algorithm with improved linear lower bounding functions*, J. Global Optim., 8 (1996), pp. 393–411.

[41] X. J. Wang and T. S. Chang, *A multivariate global optimization using linear bounding functions*, J. Global Optim., 12 (1998), pp. 383–404.

[42] J. M. Zamora and I. E. Grossmann, *A branch and contract algorithm for problems with concave univariate, bilinear and linear fractional terms*, J. Global Optim., 14 (1999), pp. 217–249.