Mathematical Programming manuscript No.
(will be inserted by the editor)

Siriporn Hongthong · R. Baker Kearfott

Rigorous Linear Overestimators and Underestimators

Received: date / Revised version: date

Abstract. For some time, convex relaxations have been used to obtain lower bounds on global optima of general nonlinear programs. One variant of this technique is to use linear relaxations, and to solve the relaxations with state-of-the-art linear programming technology. We are examining a variant of linear relaxations in which we completely (and automatically) parse the objective into component elementary operations (addition, multiplication, powers, etc.), then relax the individual operations. A number of researchers have explored and implemented these variants, but since floating point arithmetic has been used both in obtaining the relaxations and in solving the resulting convex (or linear) programs, the lower bounds on the objective may not be actual lower bounds. Recently, Neumaier and Shcherbina, as well as Jansson, have shown how to get rigorous lower bounds on the solution of linear programs, and others have begun to explore using these rigorous lower bounds in the context of convex relaxations. In particular, if a point, machine-representable linear program is presented to a process that obtains a rigorous lower bound on its optimum, then that point program should be a rigorous relaxation of the original nonlinear program, for the rigorous lower bound to be a lower bound for the original nonlinear program. In the context of relaxing the individual operations, this means supplying rigorously rounded coefficients for underestimators and overestimators for the operations. This work gives details of how the estimation and rounding can be done for odd and even powers, reciprocals, exponentials, logarithms, square roots, and uncertain scalar multiples.

1. Introduction

We consider the general global nonlinear programming problem (NLP) defined by

$$\begin{array}{l} \text{minimize } \varphi(x) \\ \text{subject to } c_i(x) = 0,\ i = 1, \ldots, m_1, \\ \qquad\qquad g_i(x) \le 0,\ i = 1, \ldots, m_2, \\ \text{where } \varphi : \boldsymbol{x} \to \mathbb{R} \text{ and } c_i, g_i : \boldsymbol{x} \to \mathbb{R}, \text{ and where } \boldsymbol{x} \subset \mathbb{R}^n \text{ is} \\ \text{the hyperrectangle (box) defined by} \\ \qquad\qquad \underline{x}_i \le x_i \le \overline{x}_i,\ 1 \le i \le n, \\ \text{where the } \underline{x}_i \text{ and } \overline{x}_i \text{ are constant bounds.} \end{array} \qquad (1)$$

The context in which we solve problem (1) is *deterministic branch and bound methods* as explained, for example, in [2,10,14]. In such methods, we adaptively

Siriporn Hongthong: University of Louisiana, Box 4-1010, Lafayette, Louisiana 70504-1010, USA, sxh1113@hotmail.com

R. Baker Kearfott: University of Louisiana, Box 4-1010, Lafayette, Louisiana 70504-1010, USA, rbk@louisiana.edu

Mathematics Subject Classification (1991): 90C26, 65G20, 65G30, 65G50

subdivide an initial region $\boldsymbol{x}^{(0)}$ into subregions $\boldsymbol{x}$ of the form in (1) while maintaining an upper bound $\overline{\varphi}$ on the global optimum of (1) (say, by evaluating at a succession of feasible points). A lower bound $\underline{\varphi}(\boldsymbol{x})$ on φ over $\boldsymbol{x}$ is computed over each subregion $\boldsymbol{x}$, and $\boldsymbol{x}$ is rejected if $\underline{\varphi}(\boldsymbol{x}) > \overline{\varphi}$.

A popular way of obtaining underestimators is through convex or linear relaxations. If the objective φ (or one or more constraints g_i) is replaced by a linear (or, more generally, a convex) function ℓ such that $\ell(x) \le \varphi(x)$ (or $\ell(x) \le g_i(x)$) for $x \in \boldsymbol{x}$, then the resulting problem has global optimum less than or equal to the global optimum of (1). If the objective and all of the constraints are underestimated by linear functions (where the equality constraints are replaced by pairs of underestimating functions), the solution to the resulting linear (or convex) program is an underestimator $\underline{\varphi}(\boldsymbol{x})$.

Techniques for linear underestimation are developed and reviewed in [14], while techniques for more general convex underestimation appear in [2].

Convex and linear underestimation techniques perhaps began with McCormick [6,7]. Also perhaps originating with McCormick was the idea of converting an arbitrary NLP into a separable one by introducing intermediate variables and replacing each elementary operation in computation of the objective and constraints by equality or inequality constraints [5][1] . In [4], we explain this process with examples, we analyze this process from the point of view of equivalency to the original NLP after replacing equality constraints by inequality constraints, and we analyze refinement of the approximations either by adding constraints or subdividing the domains $\boldsymbol{x}$. This work is within that context.

Convex or linear relaxations of nonlinear programs have been used for some time and by many in branch and bound methods for global optimization, but the linear programs themselves have both been formed approximately (with floating-point arithmetic) and solved approximately. Thus, the lower bounds $\underline{\varphi}(x)$ so obtained are only "approximate" lower bounds, with no guarantee that they are actual lower bounds. Therefore, the resulting algorithms have been not validated (that is, they give global optima and global optimizers without mathematical guarantees). Only recently, with work of Neumaier and Shcherbina [9] and of Jansson [3], has it been recognized how to obtain a rigorous lower bound, given approximate values of the dual variables. These new techniques allow us to obtain a rigorous lower bound $\underline{\varphi}(\boldsymbol{x})$, given the solution to a linear relaxation. However, such a lower bound will not be rigorous unless the linear relaxation is itself rigorous in the sense that, although the coefficients are machine representable numbers, the resulting linear functions are exactly (and not just approximately) lower bounds on the corresponding elements of the original problem.

Although taking account of rounding when computing point linear programs representing relaxations of problem (1) appears to be possible without significant increase in overall computational effort and without the necessity to discover not-already-known obscure tricks, there are numerous pitfalls to be avoided if we want the resulting linear program with machine-representable coefficients

[1] Actually, McCormick's conversion process is closely related to basic ideas in automatic differentiation, as presented, for example, in the early reference [11].

to be truly a relaxation of the original problem. Furthermore, not all methods of producing approximate relaxations are easily transformable to methods that produce rigorous relaxations. It is useful to provide details of how such rigorous relaxations can be produced. The only such works of which we are aware that precedes this paper are [1] and [8]. In [1], Borradaile and Van Hentenryck develop a short theory of rigorous underestimating and overestimating lines (and optimality thereof) for convex and concave functions, given lower and upper bounds on the slopes (linear coefficients) and y-intercepts (constant coefficients). In [8], Michel, Lebbah, and Rueher work in a context similar to that of this paper; they give explicit and detailed procedures on how to rigorously round when bounding products and quadratic terms in a quadratic program.

In this paper, we show more generally how to bound arbitrary positive integer powers of x, quotients, exponential functions, logarithms, square roots, and $\boldsymbol{a}x$, $\boldsymbol{a}$ a constant with uncertainty, over arbitrary intervals. In particular, we give details of how to rigorously underestimate (or overestimate) x^n, n odd, by a set of linear inequalities, such that the total effect of the set of underestimating (or overestimating) linear inequalities represents an approximation to a specified accuracy over the portion of the interval in which multiple underestimators are possible. We explain our suggested technique for underestimating and overestimating odd powers in §2, while we explain relaxations of even powers in §3, of exponential functions in §4, of logarithms in §5, and of square roots in §6. We describe a rational way of handling quotients in §7, and we give underestimates for $\boldsymbol{a}x$, $\boldsymbol{a}$ an uncertain (interval) coefficient in §8. We relate our work to that in [1] and [8] in §9, and give conclusions and the direction for future work in §10.

2. Relaxations of Odd Powers

2.1. Background

Let $f(x) = x^n$, $x \in [a, b]$, where n is a positive odd number and a and b are machine numbers with $a < 0$ and $b > 0$. The convex envelope and concave envelope of $f(x) = x^n$ are shown in Figure 1 and Figure 2, respectively. To find a linear underestimator of x^n, first we let $p_1 = a$ and $p_2 = b$ for the underestimator and let $p_1 = b$ and $p_2 = a$ for the overestimator. We define the function $g(x)$, the difference between the slope of the secant line connecting the points (p_1, p_1^n) and (x, x^n) and the slope of the tangent line at x for $x \in (a, b)$, by

$$g(x) = \frac{x^n - p_1^n}{x - p_1} - nx^{n-1}.$$

Then, g is monotone decreasing for the underestimator and monotone increasing for the overestimator. The solution t of $g(x) = 0$ is the point where the slope of the secant line connecting the points (p_1, p_1^n) and (t, t^n) equals the slope of the tangent line at t. Therefore, tight linear estimators of x^n, $x \in [a, b]$ are the secant

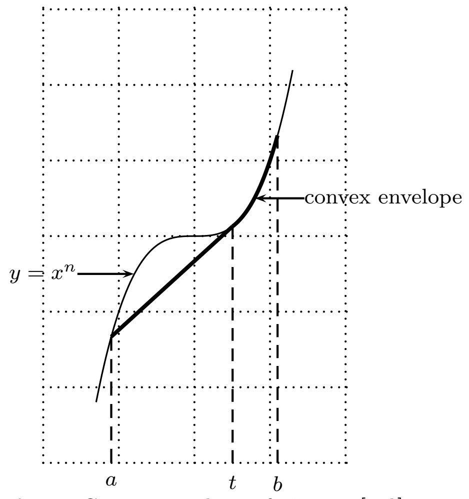


Fig. 1. Convex envelope of x^n, $x \in [a, b]$

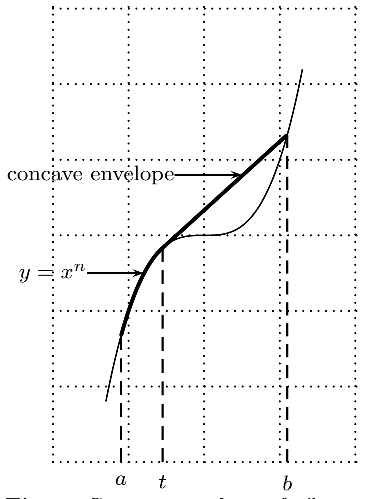


Fig. 2. Concave envelope of x^n, $x \in [a, b]$

line connecting the points (p_1, p_1^n) and (t, t^n) and linear estimators of x^n, where x is between t and p_2. Solving $g(x) = 0$ is equivalent to solving the equation

$$(1 - n)x^n + p_1 n x^{n-1} - p_1^n = 0. \tag{2}$$

Let $h(x) = (1 - n)x^n + p_1 n x^{n-1} - p_1^n$. Then $h'(x) = -n(n-1)x^{n-2}(x - p_1)$. The Newton's method iterates are defined by

$$x_{k+1}(h, x_k) = x_k - \frac{h(x_k)}{h'(x_k)},$$

where x_0 is some appropriate initial approximation.

Next, we show how to choose x_0 in order to start the iteration to converge to the solution t. Let's view Newton's method as a fixed point iteration. Let $p(x) = x - \frac{h(x)}{h'(x)}$. Then p is continuous. Assume the following two conditions hold:

1. $h(x)$ has one positive real solution for the underestimator and one negative real solution for the overestimator;
2. $p'(x) > 0$ when $x > t$ for the underestimator and $p'(x) > 0$ when $x < t$ for the overestimator.

Then Newton's method converges to the solution t. See Figure 3 and Figure 4.

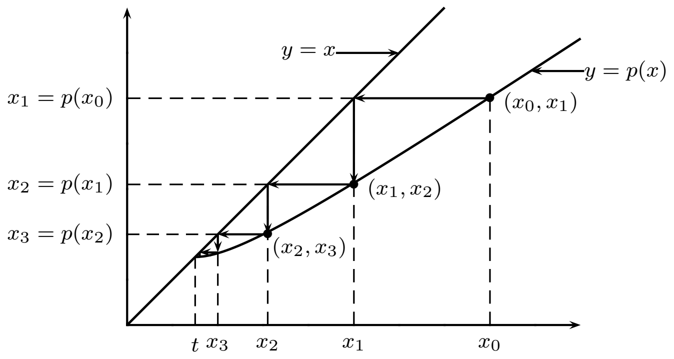


Fig. 3. Newton's method converges to t for the underestimator.

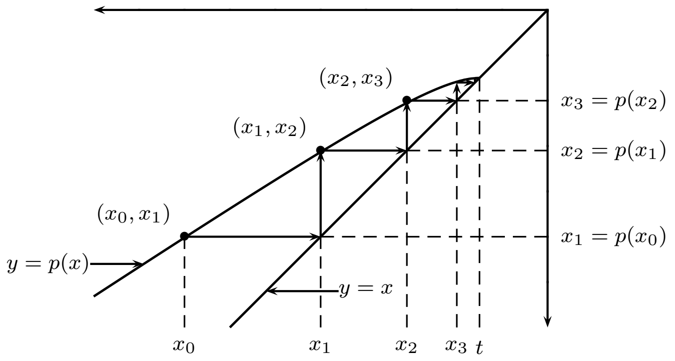


Fig. 4. Newton's method converges to t for the overestimator.

To show that the above two assumptions hold, we consider $h(x) = (1 - n)x^n + p_1 n x^{n-1} - p_1^n$ and $h''(x) = -n(n-1)^2 x^{n-2} + p_1 n(n-1)(n-2)x^{n-3}$, where n is a positive odd number. For the underestimator, $p_1 < 0$. Then $h(x)$ has one variation in sign. Thus, by Descartes's rule of signs, $h(x) = 0$ has one and only one positive real solution. Also, $h''(x) < 0$ for $x > t$. Rewriting $h(x) = g(x)(x - p_1)$, we see that h is monotone decreasing. Since $h(0) > 0$ and $h(x)$ has only one positive real solution, $h(x) < 0$ for $x > t$. Thus

$$p'(x) = \frac{h(x)h''(x)}{(h'(x))^2} > 0, \text{ when } x > t.$$

Hence, for the underestimator, we start x_0 from the right end of the interval, i.e. $x_0 = b - \epsilon$, for small ϵ. In the same manner, for the overestimator, $p_1 > 0$. Then $h(-x)$ has one variation in sign. Thus, by Descartes's rule of signs, $h(x) = 0$ has one and only one negative real solution. Also, $h''(x) > 0$ for $x < t$. Rewriting $h(x) = g(x)(x - p_1)$, we see that h is monotone decreasing. Since $h(0) < 0$ and $h(x)$ has only one negative real solution, $h(x) > 0$ for $x < t$. Thus, $p'(x) > 0$, when $x < t$. Therefore, we start x_0 from the left end of the interval, i.e. $x_0 = a + \epsilon$ for the overestimator.

However, for both the underestimator and overestimator, the given interval may not include the solution t or the solution may occur at p_1 which is not in the domain of g. In this case, the linear estimator is a secant line connecting the two endpoints.

2.2. Method to obtain rigorous estimators

The main thing is we want to get validated solutions. There are three things we need to be concerned about: the point t, the secant line from (p_1, p_1^n) to (t, t^n), and the linear estimator of x^n, where x is between t and p_2. Using the considerations of the previous section, t is computed only approximately. To use this approximation to t to find the rigorous estimators, this point itself must be validated. Therefore, we construct a box in which we know this point t has to lie. We do this by using the interval Newton method. We construct a box $\boldsymbol{x}$ centered at the approximation $\check{t}$ which we got from Newton's method. For it to be possible to verify existence or uniqueness, the box $\boldsymbol{x}$ should be larger than the tolerance which the actual solution will be computed. The interval Newton operator is

$$\boldsymbol{N}(\boldsymbol{h}, \boldsymbol{x}, \check{t}) = \check{t} - \frac{\boldsymbol{h}(\check{t})}{\boldsymbol{h}'(\boldsymbol{x})}.$$

We get an interval $[\underline{n}_2, \overline{n}_2]$. Then, for rigor, we use interval arithmetic to evaluate $f([\underline{n}_2, \overline{n}_2]) = (\underline{f}, \overline{f})$. We obtain the box as shown in Figure 5 and Figure 6. The point $(\overline{n}_2, \underline{f})$ on the bottom right corner is chosen to be t^* for the underestimator, whereas the point $(\underline{n}_2, \overline{f})$ on the top left corner is chosen to be t^* for the overestimator. A point t^* corresponding to this box will be used to find the rigorous estimators. For the secant line connecting two points, we need to make sure that it guarantees a correct estimator. For example, if $m_u x + b_u$ and

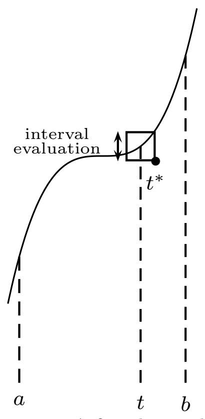


Fig. 5. A point t^* for the underestimator

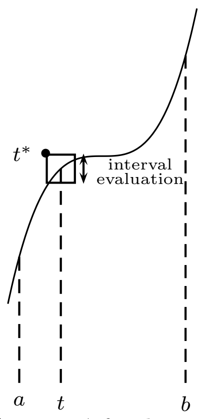


Fig. 6. A point t^* for the overestimator

$m_o x + b_o$ are the linear underestimator and linear overestimator of $f(x)$, then $m_u x + b_u \le f(x)$ and $m_o x + b_o \ge f(x)$ must be strictly valid. We will translate two pairs of coordinates representing the secant line into a machine-representable slope (linear coefficient) and y-intercept (constant coefficient). Also, for the linear estimator of x^n, where x is between t and p_2, we are going to use the adaptive method to obtain piecewise linear estimators, because a single linear estimator may not be sharp. In other words, we will subdivide the interval and produce linear approximations in each subinterval. Therefore, the rigorous estimator has two different parts:

1. a line obtained by using algorithm 2 below with the following inputs: 'is_under', [a,b], t^*.
2. piecewise linear estimators obtained by using algorithm 1 below with the following inputs: 'is_under', interval between t^* and p_2, ϵ.

'is_under' is a logic variable to determine whether we want to find underestimators or overestimators. It is 'T' for underestimators.

2.2.1. Piecewise linear estimators We use essentially the sandwich algorithm [12], except we modify the line and construction to take account of roundoff error to obtain piecewise linear estimators. Algorithm 1 is proposed for finding piecewise linear underestimators for convex functions and overestimators for concave functions. To take account of roundoff error in computing a linear function, we define the function round(is_under,sign) as in Table 1. Because of the way the

Table 1. Function round(is_under,sign) shows how to round in computing the slope (linear coefficient) and y-intercept (constant coefficient) of a linear function.

		rounding in computing	
is_under	**sign**	**slope**	**y-intercept**
T	$+$	upward	downward
T	$-$	downward	downward
F	$-$	upward	upward
F	$+$	downward	upward

constraints are formulated in the LP and validated computations, we will translate the representation in terms of two endpoints into a representation in terms of a machine-representable slope (linear coefficient) and y-intercept (constant coefficient). We define two types of rounding as follows.

Definition 1. $\lfloor x \rfloor$ *is a machine number computed with correct rounding to be less than or equal to the exact (mathematical quantity) x (and preferably the nearest machine number less than x).*

Definition 2. $\lceil x \rceil$ *is a machine number computed with correct rounding to be more than or equal to the exact (mathematical quantity) x (and preferably the nearest machine number more than x).*

Note that x may represent an expression or a function value, and actually computing $\lfloor x \rfloor$ or $\lceil x \rceil$ may involve more than one rounding operation.

We use Borradaile and Van Hentenryck 's technique [1] to define the function segment(is_under,sgn($\underline{x}$),sgn($\overline{x}$)) as in Table 2. Illustrative examples are given below.

Example 1. Find a machine-representable underestimating approximate tangent line to the curve $y = x^n$ at the point $d > 0$ by using rounding scheme round(is_under,+).

Without taking account of rounding error, the tangent line is

$$y = nd^{n-1}x - nd^n + d^n,$$

which gives the slope nd^{n-1} and y-intercept $-nd^n + d^n$. Since we are finding a linear underestimator, is_under='T'. Therefore, we use upward rounding in computing the slope and downward rounding in computing the y-intercept.

Table 2. Function segment(is_under,sgn($\underline{x}$),sgn($\overline{x}$)) gives the slope (linear coefficient) and y-intercept (constant coefficient) of a machine-representable line segment below(if is_under is 'T')/above(if is_under is 'F') the line segment $mx + b, x \in [\underline{x}, \overline{x}]$.

is_under	**sgn($\underline{x}$)**	**sgn($\overline{x}$)**	**slope**	**y-intercept**
T	$-$	$-$	$\lceil m \rceil$	$\lfloor b \rfloor$
T	$+$	$+$	$\lfloor m \rfloor$	$\lfloor b \rfloor$
T	$-$	$+$	η_1	$\lfloor b + \gamma_1 \rfloor$
F	$-$	$-$	$\lfloor m \rfloor$	$\lceil b \rceil$
F	$+$	$+$	$\lceil m \rceil$	$\lceil b \rceil$
F	$-$	$+$	η_2	$\lceil b + \gamma_2 \rceil$

where

(*i*) $\eta_1 = \eta_2 = \lfloor m \rfloor, \gamma_1 = (\lceil m \rceil - \lfloor m \rfloor)\underline{x}$, and $\gamma_2 = (\lceil m \rceil - \lfloor m \rfloor)\overline{x}$, if $|\overline{x}| \le |\underline{x}|$;

(*ii*) $\eta_1 = \eta_2 = \lceil m \rceil, \gamma_1 = -(\lceil m \rceil - \lfloor m \rfloor)\overline{x}$, and $\gamma_2 = -(\lceil m \rceil - \lfloor m \rfloor)\underline{x}$, if $|\underline{x}| \le |\overline{x}|$.

Both are applicable. The difference is in the tightness.

Even though power is a built-in function, unless we know the accuracy of the floating evaluation of this function, we either implement our own version of the power function with proper rounding or we use interval arithmetic to obtain these bounds. In our INTLAB [13] implementation, the computations are as follows:

$$\begin{aligned} d &= \text{intval}(d), \\ \text{slope} &= \sup(nd^{n-1}), \\ \text{y-intercept} &= \inf(-nd^n + d^n). \end{aligned}$$

Note that "intval" defines the type of variable to be an interval; "sup" and "inf" return the right and left bound of the interval, respectively.

Example 2. Let $x_1 < 0, x_2 > 0$, and $|x_1| < |x_2|$. Find a machine-representable slope and y-intercept for a line segment that overestimates the line segment connecting the points (x_1, y_1) and (x_2, y_2), by using the function segment('F',sgn(x_1),sgn(x_2)).

The "exact" line segment is $mx + b$, where

$$\begin{aligned} m &= \frac{y_2 - y_1}{x_2 - x_1}, \\ b &= -\frac{y_2 - y_1}{x_2 - x_1} x_1 + y_1. \end{aligned}$$

Figure 7 shows that $\lceil m \rceil x + \lceil b \rceil$ is not a correct overestimator because the large slope can cause $\lceil m \rceil x + \lceil b \rceil$ to dip below the exact value at the lower endpoint

$\underline{x}$. By using the function segment('F', sgn(x_1), sgn(x_2)), we obtain

$$\text{slope} = \left\lceil \frac{y_2 - y_1}{x_2 - x_1} \right\rceil,$$
$$\text{y-intercept} = \left\lceil -\frac{y_2 - y_1}{x_2 - x_1} x_1 + y_1 - (\lceil m \rceil - \lfloor m \rfloor) x_1 \right\rceil.$$

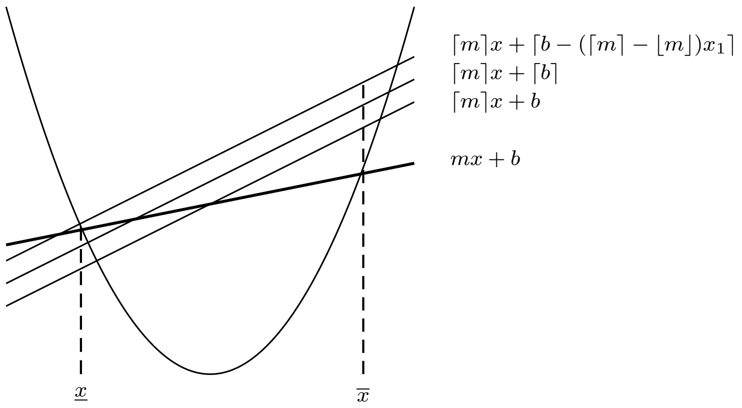

$\lceil m \rceil x + \lceil b - (\lceil m \rceil - \lfloor m \rfloor) x_1 \rceil$
$\lceil m \rceil x + \lceil b \rceil$
$\lceil m \rceil x + b$
$mx + b$
$\underline{x}$
$\overline{x}$

Fig. 7. $\lceil m \rceil x + \lceil b \rceil$ is not guaranteed to be a correct overestimator.

Piecewise linear underestimators for convex functions and piecewise linear overestimators for concave functions can be obtained from the following algorithm. The input is the logic variable is_under ('T' for underestimators and 'F' for overestimators), the interval [a,b] where a and b are machine numbers and the tolerance ϵ, while the output is machine-representable coefficients for a set of piecewise linear functions.

Algorithm 1 (Find piecewise linear estimators for $y = f(x)$, $x \in [a, b]$.)

Input: is_under, interval $[a, b]$, ϵ
Output: machine coefficients for a set of piecewise linear underestimators (if f is convex and is_under = 'T') or overestimators (if f is concave and is_under = 'F')

initialize:

```
(a) $n \leftarrow 1$.
(b) $l(n) \leftarrow a$, $r(n) \leftarrow b$.
```

do while $n > 0$*:*

```
    1. Evaluate the tangent lines to $y = f(x)$ at the endpoints of the interval $[l(n), r(n)]$ by computing the coefficients using rounding scheme round(is_under,$-$) at $l(n)$ and round(is_under,$+$) at $r(n)$. We obtain an initial bound for the function in the interval $[l(n), r(n)]$.
    2. Compute the point of intersection for the tangent lines resulting from the previous step. The abscissa of the point of intersection is called $c$. The interval $[l(n), r(n)]$ is partitioned at this point $c$.
    3. Form two new subintervals $[l(n), c]$ and $[c, r(n)]$.
    4. Calculate the maximum distance between the point $c$ to the curve $f(x)$, $x \in [l(n), r(n)]$, and let it be $d$. This maximum distance $d$ is the maximum error between the curve and its initial bound. The abscissa of the point where the maximum error occurs is called $x_1$. The "exact" bound corresponds to a tangent line to $f(x)$ at $x_1$. (see Figure 8.)
    5. Find rigorously computed slope and $y$-intercept for a bound to $f$ that is approximately tangent to $f$ at $x_1$. (see Algorithm 2 below.)
    6. If $d < \epsilon$, then
           $(a)$ Store $l(n)$, $r(n)$ and the rigorous bound in a list.
           $(b)$ $n \leftarrow n - 1$.
       otherwise assign the new left and right interval as follows:
           New left interval:
               $(a)$ $l(n) \leftarrow l(n)$.
               $(b)$ $r(n) \leftarrow c$.
           New right interval:
               $(a)$ $l(n+1) \leftarrow c$.
               $(b)$ $r(n+1) \leftarrow r(n)$.
               $(c)$ $n \leftarrow n + 1$.
       end if
end do
```

We give details of step 5 in Algorithm 2 below.

Algorithm 2 (Find rigorously computed slope and y-intercept for a bound to $f(x)$ that is approximately tangent to f at a point x_1.)

```
Input: is_under, interval $[a, b]$, $x_1$
Output: a machine-representable slope and $y$-intercept corresponding to a mathematically correct bound

    1. Calculate a slope $a_l$ and $y$-intercept $b_l$ of an approximating the tangent line to the left of $x_1$, denoted $L(x)$, by using rounding scheme round(is_under,$+$), and a slope $a_r$ and $y$-intercept $b_r$ of an approximating tangent line to the right of $x_1$, denoted $R(x)$, by using rounding scheme round(is_under,$-$). (see Figure 9.)
    2. Evaluate $L(l(n)) = a_l l(n) + b_l$ and $R(r(n)) = a_r r(n) + b_r$ by using downward rounding in the multiplication and addition in an underestimator and upward rounding in the multiplication and addition in an overestimator.
```

3. *Compute a machine-representable slope and y-intercept corresponding to the line connecting $(l(n), L(l(n)))$ and $(r(n), R(r(n)))$ by using the function segment(is_under,sgn(l(n)),sgn(r(n))). The line corresponding to this machine-representable slope and y-intercept is a rigorously computed bound, as shown in Figure 9.*

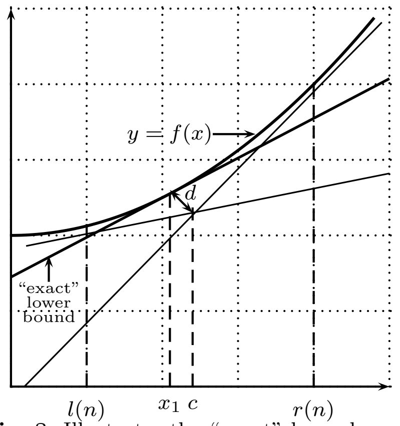


Fig. 8. Illustrates the "exact" lower bound.

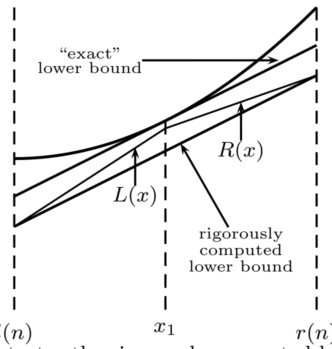
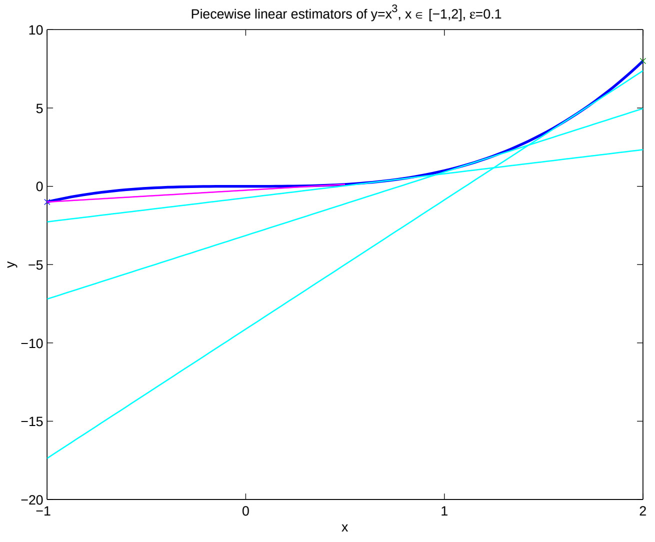


Fig. 9. Illustrates the rigorously computed lower bound.

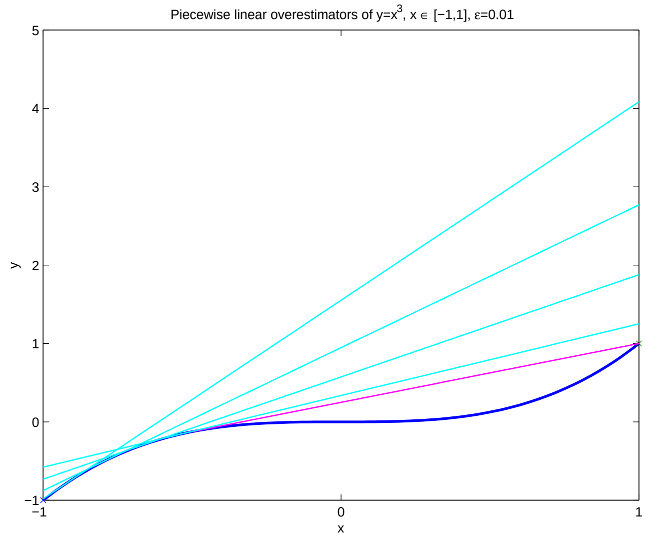
Example 3. Find the linear underestimators of $y = x^3$, where $x \in [-1, 2]$.

As in algorithm 1 with $\epsilon = 0.1$, we obtain piecewise linear underestimators shown below.



Example 4. Find the linear overestimators of $y = x^3$, where $x \in [-1, 1]$.

As in algorithm 1 with $\epsilon = 0.01$, we obtain piecewise linear overestimators shown below.



A summary of how the relaxations of the function $y = f(x), x \in [\underline{x}, \overline{x}]$, where $\underline{x}$ and $\overline{x}$ are machine numbers, depend on the convexity of the function is shown in the following table.

Function	**Underestimator(s)**	**Overestimator(s)**
Convex	use algorithm 1 with is_under =‘T’ and $[\underline{x}, \overline{x}]$ to obtain piecewise linear underestimators.	use the function segment (‘F’,sgn($\underline{x}$),sgn($\overline{x}$)) to compute a machine-representable slope and y-intercept for a line segment that overestimates the line segment connecting $(\underline{x}, \lceil f(\underline{x}) \rceil)$ and $(\overline{x}, \lceil f(\overline{x}) \rceil)$.
Concave	use the function segment (‘T’,sgn($\underline{x}$),sgn($\overline{x}$)) to compute a machine-representable slope and y-intercept for a line segment that underestimates the line segment connecting $(\underline{x}, \lfloor f(\underline{x}) \rfloor)$ and $(\overline{x}, \lfloor f(\overline{x}) \rfloor)$.	use algorithm 1 with is_under =‘F’ and $[\underline{x}, \overline{x}]$ to obtain piecewise linear overestimators.

3. Underestimators and Overestimators for Even Powers

Let $f(x) = x^n$, $x \in [\underline{x}, \overline{x}]$, where n is a positive even number and $\underline{x}$ and $\overline{x}$ are machine numbers. Then f is a convex function. In exact arithmetic, a linear underestimator is the tangent line at any point in $[\underline{x}, \overline{x}]$ and a linear overestimator is the secant line connecting the points $(\underline{x}, \underline{x}^n)$ and $(\overline{x}, \overline{x}^n)$. However, as in the case of odd powers, a rigorously computed slope and y-intercept corresponding to a lower bound can be obtained by using algorithm 2. Moreover, with the adaptive process, piecewise linear underestimators (corresponding to sets of inequalities) can be obtained by using algorithm 1. For a linear overestimator, we use the function segment(‘F’,sgn($\underline{x}$),sgn($\overline{x}$)) to obtain a machine-representable slope and y-intercept for a line segment that overestimates the secant line connecting the points $(\underline{x}, \lceil \underline{x}^n \rceil)$ and $(\overline{x}, \lceil \overline{x}^n \rceil)$. Even though power is a built-in function, unless we know the accuracy of the floating evaluation of this function, we either write our own versions with correct rounding or use interval arithmetic to obtain these estimators.

4. Underestimators and Overestimators for Exponential Functions

Let $f(x) = e^x$, $x \in [\underline{x}, \overline{x}]$, $\underline{x}$ and $\overline{x}$ machine numbers. Since f is convex, as in the case of even powers, rigorously computed coefficients for a lower bound and piecewise linear underestimator can be obtained with algorithm 2 and algorithm 1, respectively. For a linear overestimator, we use the function segment(‘F’,sgn($\underline{x}$), sgn($\overline{x}$)) to obtain a machine-representable slope and y-intercept for a line segment that overestimates the secant line connecting the points $(\underline{x}, \lceil e^{\underline{x}} \rceil)$ and $(\overline{x}, \lceil e^{\overline{x}} \rceil)$. Even though exponential is a built-in function, unless we know the accuracy of the floating evaluation of this function, we either write our own versions with correct rounding or use interval arithmetic to obtain these estimators.

5. Underestimators and Overestimators for Logarithms

Let $f(x) = \log x$, $x \in [\underline{x}, \overline{x}]$, $\underline{x}$ and $\overline{x}$ machine numbers, and $\underline{x} > 0$. Then f is a concave function. In exact arithmetic, a linear underestimator is the secant line connecting the points $(\underline{x}, \log \underline{x})$ and $(\overline{x}, \log \overline{x})$ and a linear overestimator is the tangent line at any point in $[\underline{x}, \overline{x}]$. However, as in the case of odd powers, we use the function segment('F',+,+) to obtain a machine-representable slope and y-intercept for a line segment that underestimates the secant line connecting the points $(\underline{x}, \lfloor \log \underline{x} \rfloor)$ and $(\overline{x}, \lfloor \log \overline{x} \rfloor)$. Even though log is a built-in function, unless we know the accuracy of the floating evaluation of this function, we either write our own versions with correct rounding or use interval arithmetic to obtain these estimators. For a rigorously computed upper bound and piecewise linear overestimators we use algorithm 2 and algorithm 1, respectively.

6. Underestimators and Overestimators for Square Roots

Let $f(x) = \sqrt{x}$, $x \in [\underline{x}, \overline{x}]$, $\underline{x}$ and $\overline{x}$ machine numbers and $\underline{x} \geq 0$. Since f is concave, as in the case of logarithms, we use the function segment('F',+,+) to obtain a machine-representable slope and y-intercept for a line segment that underestimates the secant line connecting the points $(\underline{x}, \lfloor \sqrt{\underline{x}} \rfloor)$ and $(\overline{x}, \lfloor \sqrt{\overline{x}} \rfloor)$. Even though square root is a built-in function, unless we know the accuracy of the floating evaluation of this function, we either write our own versions with correct rounding or use interval arithmetic to obtain these estimators. For the overestimators, we are looking for the small interval to the right of $x = 0$ such that the resulting computed $f'(x)$ will not overflow when we exclude this interval from the domain of x. Let "huge" be the largest positive real number that can be represented using IEEE standard arithmetic. Then $g'(x)$ will not overflow if $x > \frac{1}{4(\text{huge})^2}$. However, in most machines the smallest nonzero floating point number is greater than $\frac{1}{4(\text{huge})^2}$. On such machines (including IEEE arithmetic machines), we exclude only the point 0 from the domain of x. Hence, if $\underline{x} > 0$, then a rigorously computed upper bound and piecewise linear overestimators can be obtained by using algorithm 2 and algorithm 1, respectively. Otherwise, there is no overestimator.

7. Underestimators and Overestimators for Quotients

7.1. Valid Domain

Consider $g(x) = \frac{1}{x}$, $x \neq 0$. We are looking for the small interval centered at $x = 0$ such that the resulting computed estimators of $g(x)$ will not overflow when we exclude this interval from the domain of x. Let "huge" and "tiny" be the largest and smallest positive real numbers that can be represented using IEEE standard arithmetic, respectively. Then $g(x)$ will not overflow if $x > \text{tiny} > \frac{1}{\text{huge}}$ and $g'(x)$ will not overflow if $x > \frac{1}{\sqrt{\text{huge}}} > \text{tiny}$. Therefore, we exclude the interval $[-\frac{1}{\sqrt{\text{huge}}}, \frac{1}{\sqrt{\text{huge}}}]$ from the domain of x. In other words, there are neither overestimating nor underestimating constraints over any interval that intersects $[-\frac{1}{\sqrt{\text{huge}}}, \frac{1}{\sqrt{\text{huge}}}]$.

7.2. Method to obtain the rigorous estimators

We derive 3 cases of estimators that depend on $\underline{x}$ and $\overline{x}$.

case 1. $\overline{x} < -\frac{1}{\sqrt{\text{huge}}}$
For $x \in [\underline{x}, \overline{x}]$, $g(x)$ is concave. Then we use algorithm 1 to obtain piecewise linear overestimators. For the underestimator, we compute a machine-representable slope and y-intercept for a line segment that underestimates the secant line connecting the points $(\underline{x}, \lfloor g(\underline{x}) \rfloor)$ and $(\overline{x}, \lfloor g(\overline{x}) \rfloor)$ by using the function segment(‘T’,−,−).

case 2. $\underline{x} > \frac{1}{\sqrt{\text{huge}}}$
For $x \in [\underline{x}, \overline{x}]$, $g(x)$ is convex. Then we use algorithm 1 to obtain piecewise linear underestimators. For the overestimator, we compute a machine-representable slope and y-intercept for a line segment that overestimates the secant line connecting the points $(\underline{x}, \lceil g(\underline{x}) \rceil)$ and $(\overline{x}, \lceil g(\overline{x}) \rceil)$ by using the function segment(‘F’,+,+).

case 3. ($\underline{x} < -\frac{1}{\sqrt{\text{huge}}}$ and $\overline{x} > \frac{1}{\sqrt{\text{huge}}}$) or ($\underline{x} < -\frac{1}{\sqrt{\text{huge}}}$ and $\overline{x} < \frac{1}{\sqrt{\text{huge}}}$) or ($\underline{x} > -\frac{1}{\sqrt{\text{huge}}}$ and $\overline{x} > \frac{1}{\sqrt{\text{huge}}}$)
There is no underestimator and no overestimator in this case. In a branch and bound algorithm, we may wish to subdivide by trisecting to

$$\left[\overline{x}, -\frac{1}{\sqrt{\text{huge}}}\right] \bigcup \left[-\frac{1}{\sqrt{\text{huge}}}, \frac{1}{\sqrt{\text{huge}}}\right] \bigcup \left[\frac{1}{\sqrt{\text{huge}}}, \overline{x}\right].$$

8. Underestimators and Overestimators for $\boldsymbol{y} = [\underline{\boldsymbol{a}}, \overline{\boldsymbol{a}}]\boldsymbol{x}$

Various problems occurring in applications have multiplications by a number a, where a is uncertain (i.e. subject to bound constraints). An alternate way of formulating such an operation (other than as a multiplication of two variables subject to bound constraints) is as an operation

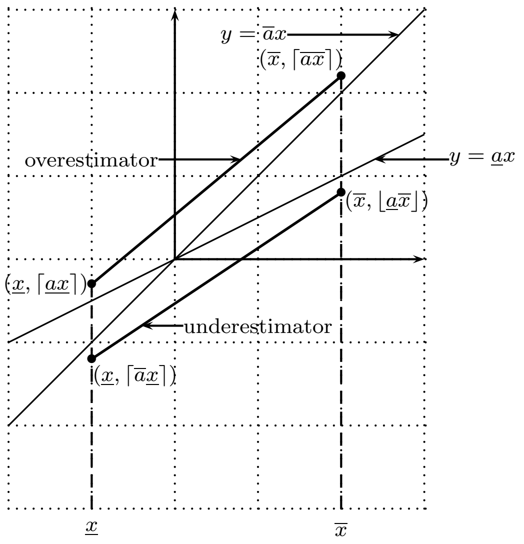
$$y = \boldsymbol{a}x,$$

where $\boldsymbol{a} \in [\underline{a}, \overline{a}]$, $\underline{a}$ and $\overline{a}$ are machine numbers and $x \in [\underline{x}, \overline{x}]$. However, if $\boldsymbol{a}$ is input as decimal and must be converted, we will use $\underline{a} \leftarrow \lfloor \underline{a} \rfloor$ and $\overline{a} \leftarrow \lceil \overline{a} \rceil$. We obtain 3 cases of underestimators and 3 cases of overestimators for y that depend on $\underline{x}$ and $\overline{x}$.

case 1. If $\overline{x} \le 0$, then
$y \ge \overline{a}x$.
$y \le \underline{a}x$.

case 2. If $\underline{x} \ge 0$, then
$y \ge \underline{a}x$.
$y \le \overline{a}x$.

case 3. If $\underline{x} \le 0$ and $\overline{x} \ge 0$, then the underestimator is given by the machine-representable slope and y-intercept for a line segment that underestimates the line segment connecting the points $(\underline{x}, \lfloor \overline{a}\underline{x} \rfloor)$ and $(\overline{x}, \lfloor \underline{a}\overline{x} \rfloor)$ computed with the function segment(‘T’,−,+) and the overestimator is given by the machine-representable slope and y-intercept for a line segment that overestimates the line segment connecting the points $(\underline{x}, \lceil \underline{a}\underline{x} \rceil)$ and $(\overline{x}, \lfloor \overline{a}\overline{x} \rceil)$ computed with the function segment(‘F’,−,+). (see Figure 10.)



$y = \overline{a}x$
$(\overline{x}, \lceil \overline{a}\overline{x} \rceil)$
overestimator
$y = \underline{a}x$
$(\overline{x}, \lfloor \underline{a}\overline{x} \rfloor)$
$(\underline{x}, \lceil \underline{a}\underline{x} \rceil)$
underestimator
$(\underline{x}, \lceil \overline{a}\underline{x} \rceil)$
$\underline{x}$
$\overline{x}$

Fig. 10. Overestimator and underestimator for $y = \boldsymbol{a}x$.

In case 3, $0 \in x$, the underestimator and overestimator of $y = \boldsymbol{a}x$ are not sharp. In branch and bound algoritms, we could improve the approximation by trisecting $\boldsymbol{x}$ in such a way that the middle interval is sufficiently small and centered at zero, that is, we trisect into $[\underline{x}, -\delta]$, $[-\delta, \delta]$ and $[\delta, \overline{x}]$, where δ is a small number.

9. Relationship to Work of Others

The implementation in section 3 is similar to [8], but we generalize to x^n. Furthermore, we find a rigorously computed bound to x^n at the point of tangency as shown in algorithm 2.

Since single linear estimators may not be sharp, we give adaptive processes for underestimators for the convex function and overestimators for the concave function as in algorithm 1 to obtain the piecewise linear estimators. The basic process has been explained in [14] and elsewhere, but we are unaware of previous details in print or of actual use in an adaptive process.

We present a technique to derive the linear underestimators and overestimators for the odd powers, as in section 2. We carefully compute the point t^* to be rigorous in order to get the rigorous linear estimators.

10. Conclusions and Future Work

We have presented techniques to produce validated linear underestimators and overestimators by taking account of roundoff error. The functions we considered are x^n, $1/x$, e^x, $\log x$ and $\sqrt{x}$. In the near future, we will also analyze rigorous relaxations of trigonometric functions such as $\sin x$ and $\cos x$, as well as other transcendental functions. Finally we are presently in the process of incorporating the work into a rigorous branch and bound method for global optimization.

References

1. G. Borradaile and P. Van Hentenryck. Safe and tight linear estimators for global optimization. In *Workshop on Interval Analysis and Constraint Propagation for Applications (IntCP 2003)*, 2003.
2. C. A. Floudas. *Deterministic Global Optimization: Theory, Algorithms and Applications.* Kluwer, Dordrecht, Netherlands, 2000.
3. Ch. Jansson. A rigorous lower bound for the optimal value of convex optimization problems. *J. Global Optim.*, 28(1):121–137, January 2004.
4. R. B. Kearfott and S. Hongthong. A preprocessing heuristic for determining the difficulty of and selecting a solution strategy for nonconvex optimization problems, 2003. preprint.
5. G. P. McCormick. Converting general nonlinear programming problems to separable nonlinear programming problems. Technical Report T-267, George Washington University, Washington, D.CThe George Washington University, 1972.
6. G. P. McCormick. Computability of global solutions to factorable nonconvex programs. *Math. Prog.*, 10(2):147–175, April 1976.
7. G. P. McCormick. *Nonlinear Programming: Theory, Algorithms, and Applications.* Wiley, 1983.
8. C. Michel, Y. Lebbah, and M. Rueher. Safe embedding of the simplex algorithm in a CSP framework. In *Proceedings of the Fifth International Workshop on Integration of AI and OR Techniques*, 2003. `http://www.crt.umontreal.ca/cpaior/article-michel.pdf`.
9. A. Neumaier and O. Shcherbina. Safe bounds in linear and mixed-integer programming. *Math. Prog.*, 2003.
10. P. M. Pardalos and J. B. Rosen. *Constrained Global Optimization: Algorithms and Applications.* Lecture Notes in Computer Science no. 268. Springer-Verlag, New York, 1987.
11. L. B. Rall. *Automatic Differentiation: Techniques and Applications.* Lecture Notes in Computer Science no. 120. Springer, Berlin, New York, etc., 1981.
12. G. Rote. The convergence rate of the sandwich algorithm for approximating convex functions. *Computing*, 48:337–361, 1992.
13. S. M. Rump et al. INTLAB home page, 2000. `http://www.ti3.tu-harburg.de/~rump/intlab/index.html`.
14. M. Tawarmalani and N. V. Sahinidis. *Convexification and Global Optimization in Continuous and Mixed-Integer Nonlinear Programming: Theory, Algorithms, Software, and Applications.* Kluwer, Dordrecht, Netherlands, 2002.