Interval Extensions of Non-Smooth Functions for Global Optimization and Nonlinear Systems Solvers

R. Baker Kearfott*
University of Southwestern Louisiana

February 29, 1996

Abstract

Most interval branch and bound methods for nonlinear algebraic systems have to date been based on implicit underlying assumptions of continuity of derivatives. In particular, much of the theory of interval Newton methods is based on this assumption. However, derivative continuity is not necessary to obtain effective bounds on the range of such functions. Furthermore, if the first derivatives just have jump discontinuities, then interval extensions can be obtained that are appropriate for interval Newton methods. Thus, problems such as minimax or l_1 approximations can be solved simply, formulated as unconstrained nonlinear optimization problems. In this paper, interval extensions and computation rules are given for the unary operation $|x|$, the binary operation $\max\{x, y\}$ and a more general "jump" function $\chi(s, x, y)$. These functions are incorporated into an automatic differentiation and code list interpretation environment. Experimental results are given for nonlinear systems involving max and $|\circ|$ and for minimax and l_1 optimization problems.

Die meisten herkömmlichen Intervallmethoden für Systeme von nichtlinearen Gleichungen und nichlineare Optimierung basieren auf die Stetigkeit der Ableitungen. Die Theorie von Intervall-Newton Methoden ist hauptsächlich auf dieser Annahme basiert. Aber Stetigkeit der Ableitungen ist nicht nötig, um wirkungsvolle Abschätzungen für den Wertebereich solcher Funktionen zu erhalten. Ausserdem können in vielen Fällen Intervallwerte gebildet werden, die für Intervall-Newton Methoden geeignet sind. So, Probleme wie l_∞ oder l_1 Annäherungen

*This work was supported in part by National Science Foundation grant CCR-9203730.

können einfacher gelöst werden, wenn sie als nichtlineares Optimierungproblem ohne Nebenbedingungen formuliert sind. Hier werden Formeln für die Berechnung von $|x|$, $\max\{x, y\}$ und eine allgemeinere "Sprungfunktion" χ gegeben. Diese Funktionen werden gebraucht im Rahmen der automatischen Diffrentiation usw. Es werden Ergebnisse von numerischen Untersuchungen für nichtlineare Systeme mit max und $|\circ|$ und für l_∞ und l_1 Optimierungsprobleme gegeben.

Key words. interval extensions, global optimization, nonsmooth optimization, nonlinear systems of equations, minimax approximation, l_1 approximation

AMS subject classifications. 65K05, 90C30, 65H10, 62C20, 90C32

1 Introduction

Branch and bound methods coupled with traditional software for finding approximate solutions, interval extensions of functions and derivatives, and coupled with interval Newton methods to accelerate the search, can form practical algorithms for rigorously finding all roots of nonlinear systems or for global optimization. For introductions to such techniques, see [4], [6], or [21], while for test results for such algorithms, see [6] or [12], [26], and others. For an advanced introduction to the underlying techniques, see [7], while for classic introductions to interval computations, see [1] or [19].

Traditionally, such methods have been implemented with subroutine packages for interval extensions, ad-hoc packages, or special commercial languages, such as the "SC" family or the "XSC" family developed at Karlsruhe ([2], [5], [18], etc.). Recent work, such as described in [4, Ch. 5], combines automatic differentiation for gradient computation to obtain Jacobi or Hessian matrices for the interval Newton Methods. The author's most recent work has used his FORTRAN-77 package INTLIB [17], combined with the Fortran 90 package described in [13] for operator overloading and automatic differentiation and slope arithmetic.

Many important problems, such as minimax, l_1 approximation, more general problems involving absolute values, or problems in which the function is defined piecewise, such as splines, are non-smooth. Special codes, such [25], or general but slow codes that do not use interval Newton methods, as [10], have been developed for non-smooth objective functions. However, except for the χ function of [13], interval extensions of functions such as max and $|\circ|$ have not been prominent in interval packages[1].

Furthermore, the theory of interval Newton methods, such as [20, Theorem 5.2.12, p. 185] usually assumes that the function components are continuously differentiable. The basic principle is that of a *slope enclosure matrix* $\mathbf{A} \subset \mathbb{R}^{n\times n}$ for the function $F : \mathbb{R}^n \to \mathbb{R}^n$, over $\mathbf{X}$ and centered at $\check{X}$:

$$\begin{array}{c}\text{For every } X \in \mathbf{X}, \text{ there exists an } A \in \mathbf{A} \text{ such that} \\ F(X) = F(\check{X}) + \mathbf{A}(X - \check{X}).\end{array} \tag{1}$$

If there is then an $X \in \mathbf{X}$ with $F(X) = 0$, we must have $X = \check{X} - A^{-1}F(\check{X})$ for some $A \in \mathbf{A}$, whence all roots of F in X must lie in the image $\check{X} - [\mathbf{A}]^I F(\check{X})$, where $[\mathbf{A}]^I F(\check{X})$ is the result of some interval Newton method; see [20]. A condition that the derivatives of F be continuous seems natural, due to the similarity of equation (1) to the mean value theorem.

Nonetheless, rigorous and effective $\mathbf{A}$ can be obtained when the derivatives of the components of F have jump discontinuities, such as when F contains terms involving $|\circ|$ or max. Interval Newton iteration can be effective in these cases, especially if extended interval arithmetic is allowed. In such cases, the computations can often sharply locate critical points corresponding to non-differentiability.

In §2, formulas for relatively sharp interval extensions for $|\circ|$, max and our function χ are given. Furthermore, formulas appropriate for symbolic differentiation of these functions, as well as for automatic differentiation (for Lipschitz matrices, cf. [20, p. 174]) and automatic slope computation appear. It is not hard to show that these formulas lead to matrices that obey the Lipschitz condition in the case of derivatives, or condition (1) in the case of slopes. Use of these formulas allows consideration of non-smooth approximation problems, etc. in the same manner as smooth problems, without special algorithms or constraints.

An example is worked in §3. Additional test examples are explained in §4, while numerical results appear in §5.

2 Formulas for Interval Extensions

These formulas fall into the following groups:

1. rules for floating-point evaluation of the operation;

[1] The function $|\circ|$ is often defined to be that floating point value termed the *magnitude*.

2. rules for floating-point evaluation of the derivative of the operation.
3. rules for interval extensions of the operation;
4. rules for symbolic differentiation of the operation;
5. rules for interval evaluation of the derivative of the operation, assuming the operation represents a continuous function, such as the absolute value;
6. rules for interval evaluation of the derivative of the operation, assuming the operation does not represent a continuous function, such as a function defined by separate formulas in separate intervals, with unmatching values at the break point;
7. rules for interval evaluation of slopes, assuming the operation represents a continuous function; and
8. rules for interval evaluation of slopes, assuming the operation does not represent a continuous function.

Here, each of these rules is presented for each of the functions χ, $|\circ|$ and max.

2.1 Formulas for $x_p = \chi(x_s, x_q, x_r)$

The function $x_p = \chi(x_s, x_q, x_r)$ was first mentioned in [13] as a device to program branches when generating code lists with operator overloading. Though not standard, as $|\circ|$ or max is, it is more general, and is used in the differentiation formulas for $|\circ|$ and max.

Formula 1 *Floating-point evaluation*

$$\chi(x_s, x_q, x_r) = \begin{cases} x_q & \textit{if } x_s < 0; \\ x_r & \textit{otherwise.} \end{cases}$$

Formula 2 *Floating-point evaluation of the derivative*

$$\frac{\partial \chi(x_s, x_q, x_r)}{\partial x_q} = \begin{cases} 1 & \textit{if } x_s < 0; \\ 0 & \textit{otherwise.} \end{cases}$$

$$\frac{\partial \chi(x_s, x_q, x_r)}{\partial x_r} = \begin{cases} 0 & \textit{if } x_s < 0; \\ 1 & \textit{otherwise.} \end{cases}$$

Formula 3 *Interval evaluation*

$$\chi(\mathbf{x}_s, \mathbf{x}_q, \mathbf{x}_r) = \begin{cases} \mathbf{x}_q & \text{if } \mathbf{x}_s < 0; \\ \mathbf{x}_r & \text{if } \mathbf{x}_s > 0; \\ \mathbf{x}_q \underline{\cup} \mathbf{x}_r & \text{otherwise.} \end{cases}$$

Formula 4 *Symbolic differentiation*

$$\chi'(x_s, x_q, x_r) = \chi(x_s, x_q', x_r')$$

Formula 5 *Interval evaluation of $\partial\chi/\partial x_q$, $\partial\chi/\partial x_r$ and $\partial\chi/\partial x_s$ when χ is continuous in x_s, i.e. when $x_q = x_r$ whenever $x_s = 0$.* (appropriate for a backward automatic differentiation process).

$$\begin{aligned} \frac{\partial\chi(\mathbf{x}_s, \mathbf{x}_q, \mathbf{x}_r)}{\partial x_q} &= \begin{cases} 1 & \text{if } \mathbf{x}_s < 0; \\ 0 & \text{if } \mathbf{x}_s > 0; \\ [0,1] & \text{otherwise.} \end{cases} \\ \frac{\partial\chi(\mathbf{x}_s, \mathbf{x}_q, \mathbf{x}_r)}{\partial x_r} &= \begin{cases} 0 & \text{if } \mathbf{x}_s < 0; \\ 1 & \text{if } \mathbf{x}_s > 0; \\ [0,1] & \text{otherwise.} \end{cases} \\ \frac{\partial\chi(\mathbf{x}_s, \mathbf{x}_q, \mathbf{x}_r)}{\partial x_s} &= 0 \end{aligned}$$

Formula 6 *Interval evaluation of $\partial\chi/\partial x_q$, $\partial\chi/\partial x_r$ and $\partial\chi/\partial x_s$ when χ is possibly discontinuous in x_s i.e. when $x_q = x_r$ whenever $x_s = 0$.* (appropriate for a backward automatic differentiation process). *The formulas are the same as formula 5 except:*

$$\begin{aligned} \frac{\partial\chi(\mathbf{x}_s, \mathbf{x}_q, \mathbf{x}_r)}{\partial x_q} &= (-\infty, \infty) \text{ if } 0 \in \mathbf{x}_s \\ \frac{\partial\chi(\mathbf{x}_s, \mathbf{x}_q, \mathbf{x}_r)}{\partial x_r} &= (-\infty, \infty) \text{ if } 0 \in \mathbf{x}_s \end{aligned}$$

Formula 7 *Interval evaluation of the slope $\mathrm{S}(\chi, \mathbf{X}, \check{\mathbf{X}})$ when χ is continuous in x_s* (appropriate for a forward automatic differentiation process).

$$\mathrm{S}(\chi(x_s, x_q, x_r), \mathbf{X}, \check{\mathbf{X}}) = \begin{cases} \mathrm{S}(x_q, \mathbf{X}, \check{\mathbf{X}}) & \text{if } \mathbf{x}_s \underline{\cup} \check{\mathbf{x}}_s < 0; \\ \mathrm{S}(x_r, \mathbf{X}, \check{\mathbf{X}}) & \text{if } \mathbf{x}_s \underline{\cup} \check{\mathbf{x}}_s > 0; \\ \mathrm{S}(x_q, \mathbf{X}, \check{\mathbf{X}}) \underline{\cup} \mathrm{S}(x_r, \mathbf{X}, \check{\mathbf{X}}) & \text{otherwise.} \end{cases}$$

Formula 8 *Interval evaluation of the slope* $\mathrm{S}(\chi, \mathbf{X}, \check{\mathbf{X}})$ *when* χ *is discontinuous in* x_s (appropriate for a forward automatic differentiation process). *The formula is the same as Formula 7 when* $0 \notin \mathbf{x}_s \underline{\cup} \check{\mathbf{x}}_s$. *When* $0 \in \mathbf{x}_s \underline{\cup} \check{\mathbf{x}}_s$, *the following is used.*

$$\mathrm{S}(\chi(x_s(X), x_q(X), x_r(X)), \mathbf{X}, \check{\mathbf{X}}) =$$
$$\begin{cases} \mathrm{S}(x_r(X), \mathbf{X}, \check{\mathbf{X}}) \underline{\cup} \left\{ \frac{1}{\mathbf{x}_s(\check{\mathbf{X}})} (\mathbf{x}_r(\mathbf{X}) - \mathbf{x}_q(\mathbf{X})) \mathrm{S}(x_s(X), \mathbf{X}, \check{\mathbf{X}}) \right\} & \text{if } \mathbf{x}_s(\check{\mathbf{X}}) > 0 \\ \mathrm{S}(x_q(X), \mathbf{X}, \check{\mathbf{X}}) \underline{\cup} \left\{ \frac{-1}{\mathbf{x}_s(\check{\mathbf{X}})} (\mathbf{x}_r(\mathbf{X}) - \mathbf{x}_q(\mathbf{X})) \mathrm{S}(x_s(X), \mathbf{X}, \check{\mathbf{X}}) \right\} & \text{if } \mathbf{x}_s(\check{\mathbf{X}}) < 0 \\ \left\{ \left[\frac{1}{\overline{x}_s(\check{\mathbf{X}})}, \infty \right) \cup \left[\frac{-1}{\underline{x}_s(\check{\mathbf{X}})}, \infty \right) \right\} (\mathbf{x}_r(\mathbf{X}) - \mathbf{x}_q(\mathbf{X})) \mathrm{S}(x_s(X), \mathbf{X}, \check{\mathbf{X}}) & \\ \qquad \underline{\cup} \mathrm{S}(x_q(X), \mathbf{X}, \check{\mathbf{X}}) \underline{\cup} \mathrm{S}(x_r(X), \mathbf{X}, \check{\mathbf{X}}) & \text{if } 0 \in \mathbf{x}_s(\check{\mathbf{X}}). \end{cases}$$

Formula 8 is explained in [15, §4] and [16]. Formula 8 is useful in optimization with interval Newton methods to enclose critical points, since the gradient of functions containing max and $|\circ|$ contains χ-expressions that are discontinuous in x_s.

2.2 Formulas for $x_p = |x_q|$

If $x \in \mathbb{R}$, then $|x| = \chi(x, -x, x)$. However, $\chi(\mathbf{x}, -\mathbf{x}, \mathbf{x})$ overestimates the range of $|\circ|$ over the interval $\mathbf{x}$, so it is advantageous to consider $|\circ|$ as a separate operation, with the following computation formulas.

Formula 9 *Floating-point evaluation of the derivative (well-known; nothing special is done at the break point)*

$$\frac{d|x_q|}{dx_q} = \begin{cases} -1 & \textit{if } x_q < 0; \\ 1 & \textit{otherwise.} \end{cases}$$

Formula 10 *Interval evaluation*

$$|\mathbf{x}| = \begin{cases} [0, \max\{|\underline{x}|, |\overline{x}|\}] & \textit{if } 0 \in \mathbf{x}; \\ [\min\{|\underline{x}|, |\overline{x}|\}, \max\{|\underline{x}|, |\overline{x}|\}] & \textit{otherwise.} \end{cases}$$

Formula 11 *Symbolic differentiation*

$$|x_q|' = \chi(x_q, -1, 1) x_q{}'$$

Formula 12 *Interval evaluation of* $d|x|/dx$ (appropriate for a backward automatic differentiation process).

$$\frac{d|\mathbf{x}_q|}{dx_q} = \begin{cases} -1 & \textit{if } \mathbf{x}_q < 0; \\ 1 & \textit{if } \mathbf{x}_q > 0; \\ [-1, 1] & \textit{otherwise.} \end{cases}$$

Formula 13 *Interval evaluation of the slope* $\mathrm{S}(|\mathbf{x}_q|, \mathbf{X}, \check{\mathbf{X}})$ (appropriate for a forward automatic differentiation process).

$$\mathrm{S}(|\mathbf{x}_q|), \mathbf{X}, \check{\mathbf{X}}) = \begin{cases} -\mathrm{S}(x_q, \mathbf{X}, \check{\mathbf{X}}) & \text{if } \mathbf{x}_q \underline{\cup} \check{\mathbf{x}}_q < 0; \\ \mathrm{S}(x_q, \mathbf{X}, \check{\mathbf{X}}) & \text{if } \mathbf{x}_q \underline{\cup} \check{\mathbf{x}}_q > 0; \\ \mathbf{S}^{(d)}(|x_q|, \mathbf{x}_q, \check{\mathbf{x}}_q)\mathrm{S}(x_q, \mathbf{X}, \check{\mathbf{X}}) & \text{otherwise,} \end{cases}$$

where

$$\mathbf{S}^{(d)}(|x|, \mathbf{x}, \check{\mathbf{x}}) = h(\underline{x}) \underline{\cup} h(\overline{x}) \quad \text{with} \quad h(x) = \begin{cases} \frac{|x|-|\check{\mathbf{x}}|}{x-\check{\mathbf{x}}} & \text{for } x \notin \check{\mathbf{x}}; \\ [-1, 1] & \text{otherwise.} \end{cases}$$

The third branch of Formula 13 is an application of a generalization of [24, Theorem 3.4]; see [15].

2.3 Formulas for $x_p = \max\{x_q, x_r\}$

For real values x_q and x_r, $\max\{x_q, x_r\} = \chi(x_r - x_q, x_q, x_r)$, but $\chi(\mathbf{x}_r - \mathbf{x}_q, \mathbf{x}_q, \mathbf{x}_r)$ overestimates the range of max for interval values $\mathbf{x}_q$ and $\mathbf{x}_r$. Formulas appropriate for max follow.

Formula 14 *Floating-point evaluation of the derivative (well-known)*

$$\frac{\partial \max\{x_q, x_r\}}{\partial x_q} = \begin{cases} 1 & \text{if } x_q > x_r; \\ 0 & \text{otherwise.} \end{cases}$$

$$\frac{\partial \max\{x_q, x_r\}}{\partial x_r} = \begin{cases} 0 & \text{if } x_q > x_r; \\ 1 & \text{otherwise.} \end{cases}$$

Formula 15 *Interval evaluation*

$$\max\{\mathbf{x}_q, \mathbf{x}_r\} = [\max\{\underline{x}_q, \underline{x}_r\}, \max\{\overline{x}_q, \overline{x}_r\}]$$

Formula 16 *Symbolic differentiation*

$$\max{}'(x_q, x_r) = \chi(x_r - x_q, x_q{}', x_r{}')$$

Formula 17 *Interval evaluation of* $\partial \max / \partial x_q$ *and* $\partial \max / \partial x_r$ (appropriate for a backward automatic differentiation process).

$$\frac{\partial \max\{\mathbf{x}_q, \mathbf{x}_r\}}{\partial x_q} = \begin{cases} 1 & \text{if } \mathbf{x}_q > \mathbf{x}_r; \\ 0 & \text{if } \mathbf{x}_q < \mathbf{x}_r; \\ [0, 1] & \text{otherwise.} \end{cases}$$

$$\frac{\partial \max\{\mathbf{x}_q, \mathbf{x}_r\}}{\partial x_r} = \begin{cases} 0 & \text{if } \mathbf{x}_q > \mathbf{x}_r; \\ 1 & \text{if } \mathbf{x}_q < \mathbf{x}_r; \\ [0, 1] & \text{otherwise.} \end{cases}$$

Formula 18 *Interval evaluation of the slope* $\mathrm{S}(\max\{\mathbf{x}_q, \mathbf{x}_r\}, \mathbf{X}, \check{\mathbf{X}})$ (appropriate for a forward automatic differentiation process).

$$\mathrm{S}(\max\{x_q, x_r\}, \mathbf{X}, \check{\mathbf{X}}) = \begin{cases} \mathrm{S}(x_q, \mathbf{X}, \check{\mathbf{X}}) & \textit{if } \mathbf{x}_q \underline{\cup} \check{\mathbf{x}}_q > \mathbf{x}_r \underline{\cup} \check{\mathbf{x}}_r; \\ \mathrm{S}(x_r, \mathbf{X}, \check{\mathbf{X}}) & \textit{if } \mathbf{x}_q \underline{\cup} \check{\mathbf{x}}_q < \mathbf{x}_r \underline{\cup} \check{\mathbf{x}}_r; \\ \mathrm{S}(x_q, \mathbf{X}, \check{\mathbf{X}}) \underline{\cup} \mathrm{S}(x_r, \mathbf{X}, \check{\mathbf{X}}) & \textit{otherwise.} \end{cases}$$

3 An Example

Consider

$$f(x) = |x^2 - x| - 2x + 2 = 0. \tag{2}$$

This function has both a root and a cusp at $x = 1$, with a left derivative of -3 and a right derivative of -1 at $x = 1$. If $1 \in \mathbf{x}$, then a slope enclosure is given by $\mathrm{S}(f, \mathbf{x}, x) = [-1, 1](\mathbf{x} + x - 1) - 2$. We will proceed with an interval Newton method

$$\begin{aligned} \tilde{\mathbf{x}} &\leftarrow x_i - f(x_i)/S(f, x_i, \mathbf{x}_i) \\ \mathbf{x}_{i+1} &\leftarrow \mathbf{x}_i \cap \tilde{\mathbf{x}}, \end{aligned}$$

with x_i equal to the midpoint of $\mathbf{x}_i$, and $\mathbf{x}_0 = [0.7, 1.1]$.

From Formula 13 and other formulas for slopes, an initial slope enclosure is computed to be

$$\mathrm{S}(f, [0.7, 1.1], .9) = [-1, 0.7][0.6, 1] - 2 = [-3, -1.3],$$

so $\tilde{\mathbf{x}} = 0.9 - \frac{0.29}{[-3,-1.3]} \subseteq [0.99\overline{6}, 1.1231]$, and $\mathbf{x}_1 = q[0.99\overline{6}, 1.1]$. Subsequent iterates are given in Table 1. Note that on iteration 3, existence was proven, since $\tilde{\mathbf{x}}_3 \subset [.9996, 1.0079]$ is strictly in the interior of $\mathbf{x}_2 \supset [.996, 1.0293]$. The width tolerance of 10^{-6} was achieved after 9 iterations. The third column of Table 1 gives approximate widths of the $\mathbf{x}_i$, and the fourth column gives ratios of successive widths of the $\mathbf{x}_i$. Thus, the convergence appears to be linear; in fact, there there appears to be a different convergence rate for the left end point than for the right end point. (Note: if monotonicity of the intermediate expression $x^2 - x$ were taken into account, so exact ranges for it were computed, then an exact inclusion for $\mathrm{S}(f, [0.7, 1.1], .9)$ could be computed, and existence could be proven on the first iteration.)

i	$\mathbf{x}_i$	width	ratios
0	[0.6999,1.1001]	4.0×10^{-1}	—
1	[0.9966,1.1001]	1.0×10^{-1}	.25
2	[0.9966,1.0293]	9.6×10^{-2}	.96
3	[0.9996,1.0079]	8.3×10^{-3}	.09
4	[0.9999,1.0022]	2.3×10^{-3}	.28
5	[0.9999,1.0007]	8.0×10^{-4}	.35
6	[0.9999,1.0001]	2.0×10^{-4}	.25

Table 1: Iterates (rounded out) and interval widths (rounded) of the interval Newton method, for the example of equation (2)

4 Test Problems and Testing Environment

We wish to test the efficacy of these operations when used to represent objective functions in global optimization and nonlinear algebraic systems. The nonlinear systems problems are somewhat easier, since only first-order derivatives are required, and the sharper formulas 5 and 7 may be used instead of 6 and 8 when computing the iteration matrix (derivative or slope) for the interval Newton methods[2].

4.1 The Testing Software and Environment

The Fortran 90 environment of [13] with the interval arithmetic package of [17] is used. The global optimization problems were tested essentially with the code of [14], while the problems involving nonlinear systems of equations were tested with the code of [12]; minor modifications had been made to these codes subsequent to the experiments in [12] and [14]. In both the optimization and nonlinear equations codes, an approximate solution was computed (if possible) first[3]. If an approximate optimum or solution was found, a box was constructed around it, and the algorithm attempted to verify existence and uniqueness (of a critical point or root) within this constructed box with ϵ-inflation [24].

[2] because symbolic differentiation of $|\circ|$ or max gives evaluations of χ that are discontinuous across the branch

[3] with `LANCELOT` for the optimization code and with `MINPACK 1` for the nonlinear equations code

In all of the test problems, a code list (ordered list of operations to evaluate the function) was first produced, within the environment of [13]. The code list was then differentiated symbolically to obtain a code list for both the objective and gradient. This derivative code list was then used in the actual optimization or nonlinear equations routines to obtain objective and gradient values, and to obtain slope matrix values when interval Newton methods were employed.

The experiments were run on a Sparc 20 with version 2.1 of the NAG Fortran 90 compiler; the debugging option, along with the lowest level of optimization, was set. Timings are given in Standard Time Units (STU's), in the context explained in [12].

4.2 Test Problems for Global Optimization

The first three problems, relatively simple, are used as an initial test of the ideas. Linear problems, they are based on fitting a line $y = ax + b$ to the data set

$$\{(x_i, y_y)\} = \{(0,1), (1,4), (2,5), (3,8)\} \tag{3}$$

in the l_2, l_1 and l_∞ sense, respectively, as follows.

l2 l_2 approximation of the simple data set (3), for comparison purposes. The function is programmed within the system [13] as follows, where the objective function value is PHI(1), where (X(I),Y(I)) is the I-th data point from (3) and where F(X) = aX+b, where a and b are the independent variables.

```
SUM = 0
DO I = 1,NDATA
   R = Y(I) - F(X(I))
   SUM = SUM + R**2
END DO
PHI(1) = SUM
```

l1 l_1 approximation of the simple data set (3). As in the previous example, the central part of the Fortran 90 code for defining this function is

```
SUM = 0
DO I = 1,NDATA
   SUM = SUM + ABS( Y(I) - F(T(I)) )
END DO
PHI(1) = SUM
```

linfty l_∞ approximation of the simple data set (3). The central part of the Fortran 90 code for it is

```
VAL = 0
DO I = 1,NDATA
   R = ABS( Y(I) - F(T(I)) )
   VAL = MAX(R,VAL)
END DO
PHI(1) = VAL
```

The above three functions were also defined in an alternate way, using the χ function exclusively. However, preliminary numerical experiments reflected the fact that this resulted in overestimates for values and derivatives.

The final two objective functions are Problem 1 and Problem 2 in [27]. Here they will be denoted by **zang1** and **zang2**.

All of these problems are two-dimensional. The starting boxes were $[-10, 10] \times [-10, 10]$ in each case.

4.3 Test Problems for Nonlinear Systems

We devised two small problems to illustrate the behavior of the software, including the interval Newton algorithms, on such systems, and to facilitate checking correctness of the coding. In the first problem, the roots do not occur at points of non-differentiability, while the roots do occur at such points in the second problem.

nle-1 This one-dimensional problem is given by

$$f(x) = |x^2 + 5x| + x + 1 = 0.$$

Its roots are at $x = -2 - \sqrt{5} \in [-4.237, -4.236]$ and $x = -3 - \sqrt{8} \in [-5.829, -5.828]$.

nle-2 This two-dimensional problem is defined by $(f_1, f_2) = (0, 0)$, where

$$\begin{aligned} f_1(X) &= \max\{\sin(x_1 + x_2), \cos(x_1 + x_2)\} \\ &\qquad - \min\{\sin(x_1 + x_2), \cos(x_1 + x_2)\} \\ f_2(X) &= |x_1| - |x_2|, \end{aligned}$$

and where $\min\{A, B\}$ was coded as $-\max\{-A, -B\}$. It has 13 solutions in $[-10, 10]$, given by $x_1 = x_2 = \pi/8 + k\pi/2$, $k = -6, -5, \ldots, 0, \ldots, 5, 6$.

5 Experimental Results

Here, we report the CPU time in standard time units (STU), the total number of boxes processed (NBOX) (not including those constructed during ϵ-inflation), the number of objective function or residual evaluations (NFUN), the number of interval Newton matrix evaluations (NMAT)[4], and, in the case of optimization, the number of gradient evaluations (NGRAD). A minimum box size tolerance (explained in [12] and [14]) of 10^{-6} was used in each case. As explained in [12], the CPU times do not reflect the minimum possible with this kind of method, since the programming environment was not meant to be optimally fast. However, they should be meaningful in relative terms; also, the total number of boxes NBOX correlates highly with the total amount of work. Floating point evaluations for the approximate optimizer or root-finder are not represented; such additional statistics are available upon request from the author.

Performance results appear in Table 2. In each case in the optimization code, the exhaustive search was successful, and the final list consisted of a single box containing the unique global optimizer. The exhaustive search also completed for both of the nonlinear equations examples. In the case of nle-1, the final list consisted of exactly two intervals, each of which was verified to contain a unique root. In the case of nle-2, the final list consisted of 13 boxes, each containing precisely one of the 13 roots of the function within the region; however, uniqueness was verified in none of the boxes.

The performance on zang1 is roughly comparable to that of the heuristic, non-interval algorithm of [27], while more effort was required for zang2. We note, however, that inclusion of non-differentiabilities is conceptually simple in this algorithm, does not require choices of parameters or smoothing functions, and leads to rigorous, exhaustive search.

Since the formulas presented here are meant to be applied as an integral part of computer codes that have been designed originally for smooth problems, a main advantage is their ease of use. That is, the formulas unify and simplify the treatment of a variety of problems. However, there is a question of how much efficiency, if any, use of the formulas offers over computer codes for verified computations that do not use derivative information. To illustrate the difference approximately, two variants of the optimization code and one variant of the nonlinear equations code were run. In the variant of the

[4] slope matrices for the Hessian matrix, in the case of optimization, and slope matrices for the Jacobi matrix, in the case of nonlinear systems

Problem	STU	NBOX	NFUN	NMAT	NGRAD
l2	0.9	5	24	2	8
l1	56.4	180	911	164	508
linfty	68.9	185	914	162	509
zang1	34.8	62	325	64	191
zang2	6005.3	6399	36742	7447	21293
nle1	0.0	2	21	29	
nle2	11.0	268	927	552	

Table 2: Performance data for non-differentiable problems

Problem	STU			NBOX		
	IN	MT	None	IN	MT	None
l2	0.9	26.4	58.0	5	152	256
l1	56.4	41.5	47.4	180	180	194
linfty	68.9	52.8	66.6	185	185	208
zang1	34.8	24.6	24.1	62	66	66
zang2	6005.3	21748.8	21778.6	6399	8403	8403
nle1	0.7	1.8		2	24	
nle2	65.5	83.1		268	867	

Table 3: Performance measures with and without derivatives

nonlinear equations code, the interval Newton method was not applied to the current box[5], so derivatives or slope information was not used to reduce the size of boxes. The first variant of the optimization code was similar: an interval Newton method (and hence slope matrices) was not used on the gradient system. In the second variant, not only was an interval Newton method not used, but gradients were not used to determine if a box could not contain critical points. (That is, the "midpoint test" was not used.)

The results for these algorithm variants appear in Table 3. There, the subcolumns labeled `IN` denote the variant with the interval Newton methods, those labelled `MT` denote the variant with the monotonicity test, but no interval Newton method,, and those labelled `NONE` represent the basic algorithm without the monotonicity test. It is seen that first and second

[5] i.e. steps 1–4 of Algorithm 6 in [12] were not done

order information are useful in reducing the total number of boxes in the nonlinear equation problems, but the second order information (corresponding to slopes across discontinuities) only appears to be effective for zang2. The CPU times are slightly more difficult to interpret, since, within the current implementation in the environment of [13], all intermediate quantities necessary for evaluation of gradients are computed whenever an objective function value is computed. However, the STU values provide roughly the same conclusions as NBOX.

Second-order information was used in all variants in the ϵ-inflation process around approximate roots. This is not reflected in the tables. However, the process was effective, since all roots or critical points except those for nle2 could be verified.

Experimental studies of a derivative-free optimization code appear in [3, 8, 9, 10], while tests of methods that involve derivatives appear in [3, 9, 11, 22, 23]. Comparisons with and without the monotonicity test appear in [3]. Further work is necessary to completely compare all techniques in these works with computer codes that incorporate extensions of non-smooth functions and their derivatives.

6 Summary

Explicit formulas for interval extensions of functions and derivatives commonly occurring in non-smooth optimization problems have been presented. With these extensions, non-smooth problems may be solved with the same algorithms as smooth problems, thus greatly simplifying the process.

References

[1] G. Alefeld and J. Herzberger. *Introduction to Interval Computations.* Academic Press, New York, 1983.

[2] J. H. Bleher, S. M. Rump, U. Kulisch, M. Metzger, C. Ullrich, and W. Walter. FORTRAN-SC — A study of a fortran extension for engineering scientific computation with access to ACRITH. *Computing,* 39(2):93–110, 1987.

[3] T. Csendes. Test results of interval methods for global optimization. In E. Kaucher, S. M. Markov, and G. Mayer, editors, *Computer Arith-*

metic, Scientific Computing, and Mathematical Modelling, pages 417–424, Basel, 1992. J. C. Baltzer AG.

[4] R. Hammer, M. Hocks, U. Kulisch, and D. Ratz. *Numerical Toolbox for Verified Computing I.* Springer-Verlag, New York, 1993.

[5] R. Hammer, M. Neaga, and D. Ratz. PASCAL-XSC, New concepts for scientific computation and numerical data processing. In E. Adams and U. Kulisch, editors, *Scientific Computing with Automatic Result Verification*, pages 15–44, New York, etc., 1993. Academic Press.

[6] E. R. Hansen. *Global Optimization Using Interval Analysis.* Marcel Dekker, Inc., New York, 1992.

[7] J. Herzberger, editor. *Topics in Validated Computations*, Studies in Computational Mathematics, Amsterdam, 1994. Elsevier Science Publishers.

[8] C. Jansson. A global optimization method using interval arithmetic. In L. Atanassova and J. Herzberger, editors, *Computer Arithmetic and Enclosure Methods*, pages 259–268, Amsterdam, Netherlands, 1992. North-Holland.

[9] C. Jansson. On self-validating methods for optimization problems. In J. Herzberger, editor, *Topics in Validated Computations*, pages 381–439, Amsterdam, Netherlands, 1994. North-Holland.

[10] C. Jansson and O. Knüppel. A global minimization method: The multidimensional case. Technical Report 92.1, Informathinstechnik, Technische Uni. Hamburg–Harburg, 1992.

[11] C. Jansson and O. Knüppel. Numerical results for a self-validating global optimization method. Technical Report 94.1, Technical University Hamburg–Harburg, February 1994.

[12] R. B. Kearfott. Empirical evaluation of innovations in interval branch and bound algorithms for nonlinear algebraic systems, 1994. Accepted for publication in *SIAM J. Sci. Comput.*

[13] R. B. Kearfott. A Fortran 90 environment for research and prototyping of enclosure algorithms for nonlinear equations and global optimization. *ACM Trans. Math. Software*, 21(1):63–78, March 1995.

[14] R. B. Kearfott. Test results for an interval branch and bound algorithm for equality-constrained optimization. In C. Floudas and P. M. Pardalos, editors, *State of the Art in Global Optimization: Computational Methods and Applications*, pages 181–200, Dordrecht, Netherlands, 1995. Kluwer.

[15] R. B. Kearfott. Treating non-smooth functions as smooth functions in global optimization and nonlinear systems solvers. In G. Alefeld and A. Frommer, editors, *Scientific Computing and Validated Numerics*, Mathematical Research, Berlin, 1995. Akademie Verlag.

[16] R. B. Kearfott. *Rigorous Branch and Bound Methods.* Kluwer, Dordrecht, Netherlands, 1996.

[17] R. B. Kearfott, M. Dawande, K.-S. Du, and C.-Y. Hu. Algorithm 737: INTLIB, a portable FORTRAN 77 interval standard function library. *ACM Trans. Math. Software*, 20(4):447–459, December 1994.

[18] C. Lawo. C-XSC – a programming environment for verified scientific computing and numerical data processing. In E. Adams and U. Kulisch, editors, *Scientific Computing with Automatic Result Verification*, pages 71–86, New York, etc., 1993. Academic Press.

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

[20] A. Neumaier. *Interval Methods for Systems of Equations.* Cambridge University Press, Cambridge, England, 1990.

[21] H. Ratschek and J. Rokne. *New Computer Methods for Global Optimization.* Wiley, New York, 1988.

[22] D. Ratz. *Automatische Ergebnisverifikation bei globalen Optimierungsproblemen.* PhD thesis, Universität Karlsruhe, 1992.

[23] D. Ratz. An inclusion algorithm for global optimization in a portable PASCAL-XSC implementation. In L. Atanassova and J. Herzberger, editors, *Computer Arithmetic and Enclosure Methods*, pages 329–338, Amsterdam, Netherlands, 1992. North-Holland.

[24] S. M. Rump. Verification methods for dense and sparse systems of equations. In J. Herzberger, editor, *Topics in Validated Computations*, pages 63–135, Amsterdam, 1994. Elsevier Science Publishers.

[25] Z. Shen, A. Neumaier, and M. C. Eiermann. Solving minimax problems by interval methods. *BIT*, 30:742–751, 1990.

[26] G. W. Walster, E. R. Hansen, and S. Sengupta. Test results for a global optimization algorithm. In P. T. Boggs, R. H. Byrd, and R. B. Schnabel, editors, *Numerical Optimization 1984*, pages 272–287, Philadelphia, 1985. SIAM.

[27] I. Zang. A smoothing-out technique for min-max optimization. *Math. Prog.*, 19(1):61–77, 1980.