An interval algorithm for constrained global optimization

M.A. Wolfe

Department of Mathematical and Computational Sciences, University of St Andrews, St Andrews, Fife KY16 9SS, United Kingdom

Received 13 May 1992; revised 3 November 1992

Abstract

An interval algorithm for bounding the solutions of a constrained global optimization problem is described. The problem functions are assumed only to be continuous. It is shown how the computational cost of bounding a set which satisfies equality constraints can often be reduced if the equality constraint functions are assumed to be continuously differentiable. Numerical results are presented.

Key words: Constrained global optimization; Interval mathematics

1. Introduction

Let \( f : \mathbb{R}^n \rightarrow \mathbb{R} \), \( g_i : \mathbb{R}^n \rightarrow \mathbb{R} \), \( i = 1, \ldots, n_g \), and \( h_j : \mathbb{R}^n \rightarrow \mathbb{R} \), \( j = 1, \ldots, n_h \), be continuous functions in a convex set \( D \subseteq \mathbb{R}^n \). Let \( X \subseteq D \) be defined by

\[
X = \{ x = (x_i)_{i=1}^n \in \mathbb{R}^n | l_i \leq x_i \leq u_i, \ i = 1, \ldots, n \} \tag{1.1}
\]

and let

\[
S = \{ x \in X | g_i(x) \leq 0, \ i = 1, \ldots, n_g, h_j(x) = 0, \ j = 1, \ldots, n_h \}. \tag{1.2}
\]

The purpose of this paper is to describe an interval arithmetic algorithm for bounding the solution set \( X_1^* \) of the global optimization problem \( P.1 \) defined by

\[
\min_{x \in S} f(x), \tag{1.3}
\]

by bounding the solution set \( X_2^* \) of the unconstrained global optimization problem \( P.2 \) defined by

\[
\min_{x \in X} F(x, c), \tag{1.4}
\]
where $F : D \times \mathbb{R}^1 \to \mathbb{R}^1$ is defined by
\[
F(x, c) = f(x) + cp(x),
\]
in which $p : D \to \mathbb{R}^1$ is defined by
\[
p(x) = \sum_{i=1}^{n_g} \max\{0, g_i(x)\} + \sum_{j=1}^{n_h} |h_j(x)|,
\]
and $c > 0$ is sufficiently large. If $x^*$ is the unique solution of P.1 and $x(c)$ is a solution of P.2, then $x(c) \to x^*, c \to \infty [4]$. However, computational experience indicates that, provided that $c > \bar{c}$, where
\[
\bar{c} = O\left(\max_{x \in X} \{|f(x)|\}\right),
\]
the interval algorithm which is described in Section 5 bounds $X^*_1$, and appears to be insensitive to the value of $c$. Indeed, as shown in [4], $\exists \bar{c} > 0$ such that $(\forall c \geq \bar{c}) x(c) = x^*$. A method for determining $\bar{c}$ to replace the empirical rule (1.7) is currently under investigation. The numerical results reported in Section 6 are obtained by using (1.7).

2. Interval mathematics

The interval mathematics which is required in order to understand this paper can be found in [1,5,7,10]. The notation which is needed is as follows. A real interval $x = [x_1, x_S]$ has infimum $x_1 \in \mathbb{R}$ and supremum $x_S \in \mathbb{R}$ with $x_1 \leq x_S$. The set of real intervals is denoted by $I(\mathbb{R})$. If $X \subseteq \mathbb{R}$, then $I(X) = \{x \in I(\mathbb{R}) | x \subseteq X\}$. The width $\text{wid}(x)$, magnitude $|x|$ and midpoint $\text{mid}(x)$ of $x \in I(\mathbb{R})$ are defined by $\text{wid}(x) = x_S - x_1$, $|x| = \max\{|x_1|, |x_S|\}$ and $\text{mid}(x) = \frac{1}{2}(x_1 + x_S)$, respectively.

A real interval vector $x = (x_{i,j})_{n_1 \times 1} \in I(\mathbb{R}^n)$ has infimum $x_1 = (x_{i,j})_{n_1 \times 1} \in \mathbb{R}^n$, supremum $x_S = (x_{i,j})_{n_1 \times 1} \in \mathbb{R}^n$, width $\text{wid}(x) = (\text{wid}(x_{i,j}))_{n_1 \times 1} \in \mathbb{R}^n$, magnitude $|x| = (|x_{i,j}|)_{n_1 \times 1} \in \mathbb{R}^n$ and midpoint $\text{mid}(x) = (\text{mid}(x_{i,j}))_{n_1 \times 1} \in \mathbb{R}^n$. The interval vector $x \in I(\mathbb{R}^n)$ represents a closed rectangular region (a box) in $\mathbb{R}^n$. It is sometimes convenient to use the alternative notation $\inf(x) = x_1$, $\sup(x) = x_S$, where $x \in I(\mathbb{R})$ and $w(x) = \max\{\text{wid}(x_i) | i = 1, \ldots, n\}$ where $x \in I(\mathbb{R}^n)$.

A real interval matrix $A = (a_{i,j})_{n \times n} \in I(\mathbb{R}^{n \times n})$ has infimum $A_1 = (a_{i,j})_{n \times n} \in \mathbb{R}^{n \times n}$, supremum $A_S = (a_{i,j})_{n \times n} \in \mathbb{R}^{n \times n}$, width $\text{wid}(A) = (\text{wid}(a_{i,j}))_{n \times n} \in \mathbb{R}^{n \times n}$, magnitude $|A| = (|a_{i,j}|)_{n \times n} \in \mathbb{R}^{n \times n}$ and midpoint $\text{mid}(A) = (\text{mid}(a_{i,j}))_{n \times n} \in \mathbb{R}^{n \times n}$. The expression $\partial_j h_i(x)$ denotes the partial derivative of $h_i$ with respect to $x_j$.

3. Inclusion functions for $p$ and $F$

Let $g_i : I(D) \to I(\mathbb{R})$, $i = 1, \ldots, n_g$, and $h_j : I(D) \to I(\mathbb{R})$, $j = 1, \ldots, n_h$, be continuous inclusion isotonic interval extensions of $g_i : D \to \mathbb{R}$, $i = 1, \ldots, n_g$, and $h_i : D \to \mathbb{R}$, $j = 1, \ldots, n_h$,
respectively. Let \( \tilde{h}_j : I(D) \to I(\mathbb{R}) \), \( j = 1, \ldots, n_h \), be defined by

\[
\tilde{h}_j(x) = \begin{cases} -h_j(x), & \sup(h_j(x)) < 0, \\ [0, \inf(h_j(x))] & \inf(h_j(x)) \leq 0 \leq \sup(h_j(x)), \\ h_j(x), & \text{otherwise.} \end{cases}
\]  

(3.1)

Then \( (\forall x \in x \in I(D)) \ |h_j(x)| \in \tilde{h}_j(x) \). Similarly, if \( \tilde{g}_i : I(D) \to I(\mathbb{R}) \), \( i = 1, \ldots, n_g \), are defined by

\[
\tilde{g}_i(x) = \begin{cases} \max\{0, \inf(g_i(x))\}, & \max\{0, \sup(g_i(x))\} \\ g_i(x), & \text{otherwise.} \end{cases}
\]  

(3.2)

then \( (\forall x \in x \in I(D)) \max\{0, g_i(x)\} \in \tilde{g}_i(x) \). Therefore, if \( p \) is defined by (1.6), then \( p : I(D) \to I(\mathbb{R}) \), defined by

\[
p(x) = \sum_{i=1}^{n_g} \tilde{g}_i(x) + \sum_{j=1}^{n_h} \tilde{h}_j(x),
\]

(3.3)

is a continuous inclusion isotonic interval extension of \( p : D \to \mathbb{R} \). Therefore, a continuous inclusion isotonic interval extension \( F : I(D) \times I(\mathbb{R}) \to I(\mathbb{R}) \) of \( F : D \times \mathbb{R} \to \mathbb{R} \) is defined by

\[
F(x, c) = f(x) + c\tilde{p}(x),
\]  

(3.4)

where if \( p \) is defined by (1.6), then \( p \) is defined by (3.3).

4. Bounding sets defined by equality constraints

As pointed out in [8], it is usually impossible to establish without doubt that \( h_i(x) = 0 \) where \( x \in x \) however narrow \( x \) is because of rounding error, even when interval arithmetic is used; but if \( \epsilon_E > 0 \), then it can be established with complete computational rigour using outwardly-rounded machine interval arithmetic that

\[
h_i(x) \subseteq [-\epsilon_E, \epsilon_E],
\]  

(4.1)

provided that \( w(x) \) is sufficiently small. It is, however, possible to establish rigorously that \( g_i(x) \leq 0, \forall x \in x \), by checking that \( \sup(g_i(x)) \leq 0 \). As remarked in [8], a relaxation requirement similar to (4.1), namely \( g_i(x) \subseteq [-\epsilon_i, \epsilon_i] \) where \( \epsilon_i > 0 \), may, if desired, be used for inequality constraints. The numerical results reported in Section 6 correspond to \( \epsilon_1 = 0 \), so that the inequality constraints are satisfied with complete computational rigour.

If only \( h_i \) is known, then the only way to determine a sub-box \( x \) of \( \bar{x} \in I(\mathbb{R}^n) \) such that (4.1) holds is by repeatedly bisecting sub-boxes of \( \bar{x} \) and rejecting those sub-boxes \( x \) such that \( 0 \notin h_i(x) \). This is usually very computationally expensive. Neumaier [6] has described a simple interval algorithm for bounding the set \( \Sigma = \{ x \in x \mid F(x) = 0 \} \) where \( F : D \subseteq \mathbb{R}^m \to \mathbb{R}^m \), \( m \leq n \), is a given mapping, with \( F \in C^1(D) \) and \( x \in I(D) \). Computational experience indicates that Neumaier’s procedure can sometimes give a considerable reduction in computational cost when used to bound the set \( H = \{ x \in x \mid h_i(x) = 0, i = 1, \ldots, n_h \} \). If \( h_i \in C^1(D), x \in I(D), z \in x \) and \( \bar{x} \in x \) such that \( h_i(\bar{x}) = 0 \), then \( \exists \xi^{(i)} \in x \) such that

\[
0 = h_i(z) + \sum_{j=1}^{n} \partial_j h_i(\xi^{(i)})(\bar{x}_j - z_j).
\]  

(4.2)
Let $\vec{d} \in \mathbb{R}^{n+1}$ be defined by

$$
\vec{d}_j = \begin{cases} 
\bar{x}_j - z_j, & j = 1, \ldots, n, \\
1, & j = n + 1,
\end{cases}
$$

(4.3)

and let $A = (a_{ij}) \in \mathbb{R}^{nh \times (n+1)}$ be defined by

$$
a_{ij} = \begin{cases} 
\partial h_i(z), & i = 1, \ldots, n_h, \ j = 1, \ldots, n, \\
\partial h_i(z), & i = 1, \ldots, n_h, \ j = n + 1.
\end{cases}
$$

(4.4)

Let $\partial h_i : \mathbb{I}(D) \to \mathbb{I}(\mathbb{R})$ be a continuous inclusion isotonic interval extension of $\partial h_i : D \to \mathbb{R}$, $i = 1, \ldots, n_h, \ j = 1, \ldots, n$. Let $x \in \mathbb{I}(D)$, let $d \in \mathbb{I}(\mathbb{R}^{n+1})$ be defined by

$$
d_j = \begin{cases} 
x_j - z_j, & j = 1, \ldots, n, \\
1, & j = n + 1,
\end{cases}
$$

(4.5)

and let $A = (a_{ij}) \in \mathbb{I}(\mathbb{R}^{nh \times (n+1)})$ be defined by

$$
a_{ij} = \begin{cases} 
\partial h_i(x), & i = 1, \ldots, n_h, \ j = 1, \ldots, n, \\
h_i(z), & i = 1, \ldots, n_h, \ j = n + 1.
\end{cases}
$$

(4.6)

If $\exists \bar{x} \in x$ such that $h_i(\bar{x}) = 0, i = 1, \ldots, n_h$, then by (4.2)–(4.6) and the inclusion isotonicity of interval arithmetic $0 \in Ad$. Therefore, if $0 \notin Ad$, then $x$ is infeasible. If $0 \in Ad$, then it is possible that $x$ contains points $\bar{x}$ such $h_i(\bar{x}) = 0, i = 1, \ldots, n_h$. By (4.2)–(4.4) for $i = 1, \ldots, n_h, \ k = 1, \ldots, n,$

$$
a_{ik} a_j = - \left\{ \sum_{j=1}^{k-1} a_{ij} \bar{d}_j + \sum_{j=k+1}^{n} a_{ij} \bar{d}_j + a_{i,n+1} \right\}.
$$

(4.7)

Also, it may be shown [7] that if

$$
\Gamma(a, b, c) = \begin{cases} 
(b/a) \cap c, & 0 \notin a, \\
I(c - (b_1/a_1, b_1/a_1)), & 0 \in a \wedge 0 < b_1, \\
I(c - (b_1/a_1, b_1/a_1)), & 0 \in a \wedge b < 0, \\
c, & 0 \in a \wedge 0 \in b,
\end{cases}
$$

(4.8)

where the interval hull $I : \mathbb{I}(\mathbb{R}) \times \mathbb{I}(\mathbb{R}) \to \mathbb{I}(\mathbb{R})$ is defined by

$$
I(u, v) = [\min\{u_1, v_1\}, \max\{u_1, v_1\}],
$$

then

$$
\Gamma(a, b, c) = I((c \in c | \exists a \in a \wedge \exists b \in b, ac = b)).
$$

Therefore, by (4.5)–(4.7) and the inclusion isotonicity of interval arithmetic, for $i = 1, \ldots, n_h, \ k = 1, \ldots, n, \ \bar{x}_k \in a_{ik} \bar{d}_k^{(i)} + z_k$ where $\bar{d}^{(i)} = d$, and for $i = 1, \ldots, n_h, \ k = 1, \ldots, n,$

$$
\bar{d}_k^{(i+1)} = \Gamma \left\{ a_{ik}, - \left\{ \sum_{j=1}^{k-1} a_{ij} \bar{d}_j^{(i)} + \sum_{j=k+1}^{n} a_{ij} \bar{d}_j^{(i)} + a_{i,n+1} \right\}, \bar{d}_k^{(i)} \right\}.
$$

(4.9)
Neumaier [6] has indicated how a formula similar to (4.9) may be used to bound \( \Sigma \) with computational cost \( O(n) \) for each value of \( i \). The determination of \( \tilde{d}^{(i+1)}_k, \ i = 1, \ldots, n_h, \ k = 1, \ldots, n, \) from (4.9) is referred to in Section 5 as a Gauss–Seidel sweep.

If \( w(x) \) is sufficiently small and \( z = \text{mid}(x) \), then \( \tilde{d}^{(n_h+1)} + z \) defined by (4.9) is often found to be a considerably sharper enclosure of \( H \) than is \( x \). The \( \delta_j h_j \) may be computed either by programming explicit formulae or by using interval derivative arithmetic or interval slope arithmetic [7,9].

5. An interval algorithm

The solution set \( X_1^* \) of P.1 may be bounded using [8, Chapter 5, Algorithm 2] (henceforth referred to as algorithm A.1). However, A.1 is computationally expensive, as exemplified by numerical results in Section 6 of this paper. Computational experience indicates that it is preferable to bound \( X_1^* \) by bounding the solution set \( X_2^* \) of P.2, with \( c \geq \bar{c} \) where \( \bar{c} > 0 \) is sufficiently large. The algorithm A.2 for bounding \( X_2^* \) which is proposed in this paper is based on this idea. The algorithm A.2, although computationally expensive, is much less so than A.1, especially for problems with equality constraints, for which the ideas described in Section 4 are applicable.

In algorithm A.2 the initial box \( x = X \) is subjected to an optional preprocessing procedure, the purpose of which is either to determine a feasible point or to bound the feasible set \( S \) for P.1. In the preprocessing procedure the feasibility of the initial box \( x \) is investigated by determining \( \text{feasible}, \text{infeasible} \in \{\text{true, false}\} \) such that

\[
\text{(feasible = true) } \iff (g_i(x) \leq \epsilon_1, \ i = 1, \ldots, n_g, \land h_j(x) \subseteq [-\epsilon_E, \epsilon_E], \ j = 1, \ldots, n_h)
\]

and

\[
\text{(infeasible = true) } \iff (\exists i \in \{1, \ldots, n_g\}, 0 < g_i(x) \land \exists j \in \{1, \ldots, n_h\}, 0 \notin h_j(x)).
\]

An upper bound \( F_U \) on the value \( f^* \) of \( f \) at each point in \( X_2^* \) is then updated from its initial value \( F_U = +\infty \) according to \( F_U = \min(F_U, \sup(f(\text{mid}(x)))) \) if \( \text{feasible} = \text{true} \), and \( F_U = \min(F_U, \sup(F(x, c))) \) if \( \text{feasible} = \text{false} \) and \( \text{infeasible} = \text{false} \).

The box \( x \) is then enqueued in a queue \( Q_p \) (initially empty) of sub-boxes of \( x \) which are to be processed further. The queue \( Q_p \) is then processed by repeated bisection, and if \( n_h > 0 \), by applying a Gauss–Seidel sweep (\( gs = 1 \)) or not (\( gs = 0 \)), producing a queue \( Q_F \) of feasible sub-boxes of \( x \) and the queue \( Q_P \) which may still contain sub-boxes of \( x \), the feasibility of which is uncertain.

The procedure which produces \( Q_F \) and \( Q_P \) is terminated either when \( (Q_P = \emptyset \lor Q_F \neq \emptyset \lor n_p > n_{p_{\max}}) \) (\( sc = 1 \)) where \( n_p \) is the number of bisections of \( x \) and \( n_{p_{\max}} \) is a given upper bound on \( n_p \), or when \( (Q_p = \emptyset \land n_p > n_{p_{\max}}) \) (\( sc = 2 \)), or otherwise when \( \max(w(y) \mid y \in L) \leq \epsilon_p \) where \( \epsilon_p > 0 \) is given, and \( L \) is a doubly-linked list (initially empty).

If, after termination of the preprocessing procedure, \( Q_p = \emptyset \) and \( Q_F = \emptyset \), then \( x \) is infeasible and the algorithm terminates. Otherwise the sub-boxes of \( x \) in \( Q_p \) and in \( Q_F \) are transferred to the list \( L \) so that the boxes \( y \in L \) are in nonincreasing order of \( w(y) \). The list \( L \) is then
processed repeatedly until either \( L = \emptyset \) or \( n_B > n_{B_{\text{max}}} \) where \( n_B \) is the number of bisections of the boxes in \( L \) and \( n_{B_{\text{max}}} \) is a given upper bound on \( n_B \), or until \( \max \{ w(y) \mid y \in L \} \leq \varepsilon_C \) where \( \varepsilon_C > 0 \) is given, as follows.

The first box \( y \) in \( L \) is removed from \( L \) and is bisected along the first edge of greatest length to obtain \( y_1 \) and \( y_2 \) such that \( y_1 \cup y_2 = y \). The boxes \( y_i, i = 1, 2 \), are then processed as follows.

The feasibility of \( y_i \) is determined. If \( \text{infeasible} = \text{true} \), then \( y_i \) is deleted. Otherwise, if \( \text{feasible} = \text{true} \), then \( F_U \) is updated according to \( F_U = \min \{ F_U, \sup(f(\text{mid}(y_i))) \} \) or, if \( \text{feasible} = \text{false} \), according to \( F_U = \min \{ F_U, \sup(F(y_i, c)) \} \). After \( F_U \) is updated, \( y_i \) is inserted into \( L \) in such a way that the boxes \( y \in L \) are in nonincreasing order of \( w(y) \).

If, after \( y_1 \) and \( y_2 \) have been processed, \( L \neq \emptyset \), then a lower bound \( F_L \) on \( f^* \) is determined according to \( F_L = \min \{ \inf(F(y, c)) \mid y \in L \} \), and all boxes \( y \) such that \( F_U < \inf(F(y, c)) \) are removed from \( L \). The algorithm terminates either when \( x = X \) has been shown to be infeasible, in which case \( X^*_2 = \emptyset \), or when \( \max \{ w(y) \mid y \in L \} \leq \varepsilon_C \).

6. Numerical results

Numerical results from Sun Pascal implementations of the algorithms A.1 and A.2 for the following examples are reported in this section.

**Example 1** (Gould [3]).

Minimize \( f(x) = (x_1 - 10)^3 + (x_2 - 20)^3 \),
subject to \( x \in ([14, 15], [0.8, 0.9]) \),
\( g_1(x) = 13 - x_1 \leq 0 \),
\( g_2(x) = 100 - (x_1 - 5)^2 - (x_2 - 5)^2 \leq 0 \),
\( g_3(x) = (x_1 - 6)^2 + (x_2 - 5)^2 - 82.81 \leq 0 \),
\( g_4(x) = -x_2 \leq 0 \).

**Example 2** (Bracken and McCormick [2]).

Minimize \( f(x) = (x_1 - 2)^2 + (x_2 - 1)^2 \),
subject to \( x \in ([0.5, 1], [0.5, 1]) \),
\( g_1(x) = \frac{1}{4}x_1^2 + x_2^2 - 1 \leq 0 \), \( h_1(x) = x_1 - 2x_2 + 1 = 0 \).

**Example 3.**

Minimize \( f(x) = (x_1 - 1)^4 + (x_2 - 1)^4 + (x_3 - 2)^4 \),
subject to \( x \in ([0, 1.1], [0, 1.1], [0, 2]) \),
\( g_1(x) = -x_1 \leq 0 \), \( g_2(x) = -x_2 \leq 0 \),
\( h_1(x) = x_1^2 + x_2^2 + x_3^2 - 6 = 0 \), \( h_2(x) = x_1^2 + x_2^2 - x_3 = 0 \).
In Tables 1–3, \( n_f \), \( n_g \), \( n_h \), and \( n_{hid} \) denote the numbers of objective, inequality constraint, equality constraint, and equality constraint derivative function evaluations on termination of A.1 or of A.2, and \( n_P \) and \( n_B \) denote the numbers of preprocessing and processing bisections respectively. Also, \( n_b \) denotes the number of boxes which remain in the list \( L \) on termination of A.1 or of A.2. Algorithms A.1 and A.2 both terminate if \( \max\{w(y) \mid y \in L\} < \epsilon_c \), with \( \epsilon_c = 10^{-3} \). Also, where preprocessing is used in A.2, \( \epsilon_P = 10^{-2} \). Cases (a)–(d) correspond to the results that are obtained using A.1 and A.2 with no preprocessing, A.2 with preprocessing (\( sc = 2, gs = 0 \)), and A.2 with preprocessing (\( sc = 2, gs = 1 \)), respectively. Clearly, A.2 with preprocessing using Gauss–Seidel sweeps (case (d)) is a drastic improvement over A.1.

Computational experience indicates that A.2 is relatively insensitive to the value of \( c \), provided that \( c = O(\|f(x)\|) \), even if \( x \) is not preprocessed so as to bound the feasible set \( S \). A theoretical basis for this empirical finding is currently under investigation. The values of \( c \) which were used are indicated in Tables 1–3. With these values of \( c \) and \( \epsilon_1 = 0 \) it is found that for problems which contain only inequality constraints \( X^*_1 \subseteq \bigcup_{y \in L} y \), so that the solution set of P.1 is bounded with complete computational rigour. For problems which contain equality constraints it is sufficient to set \( \epsilon_1 = 0 \) if inequality constraints are present but one must have \( \epsilon_E > 0 \) to bound the feasible set \( S \). In the results which are reported in this section \( \epsilon_E = 10^{-2} \).

### Table 1
Results for Example 1

<table>
<thead>
<tr>
<th>Case</th>
<th>( c )</th>
<th>( n_f )</th>
<th>( n_g )</th>
<th>( n_h )</th>
<th>( n_{hid} )</th>
<th>( n_P )</th>
<th>( n_B )</th>
<th>( n_b )</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a)</td>
<td>/</td>
<td>1357</td>
<td>2141</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>675</td>
<td>82</td>
</tr>
<tr>
<td>(b)</td>
<td>1000</td>
<td>342</td>
<td>1112</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>313</td>
<td>11</td>
</tr>
<tr>
<td>(c)</td>
<td>1000</td>
<td>186</td>
<td>1184</td>
<td>0</td>
<td>0</td>
<td>188</td>
<td>147</td>
<td>11</td>
</tr>
</tbody>
</table>

### Table 2
Results for Example 2

<table>
<thead>
<tr>
<th>Case</th>
<th>( c )</th>
<th>( n_f )</th>
<th>( n_g )</th>
<th>( n_h )</th>
<th>( n_{hid} )</th>
<th>( n_P )</th>
<th>( n_B )</th>
<th>( n_b )</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a)</td>
<td>/</td>
<td>635</td>
<td>142</td>
<td>529</td>
<td>0</td>
<td>0</td>
<td>301</td>
<td>8</td>
</tr>
<tr>
<td>(b)</td>
<td>10</td>
<td>303</td>
<td>145</td>
<td>320</td>
<td>0</td>
<td>0</td>
<td>194</td>
<td>8</td>
</tr>
<tr>
<td>(c)</td>
<td>10</td>
<td>282</td>
<td>147</td>
<td>522</td>
<td>0</td>
<td>161</td>
<td>134</td>
<td>8</td>
</tr>
<tr>
<td>(d)</td>
<td>10</td>
<td>115</td>
<td>71</td>
<td>117</td>
<td>26</td>
<td>26</td>
<td>45</td>
<td>7</td>
</tr>
</tbody>
</table>

### Table 3
Results for Example 3

<table>
<thead>
<tr>
<th>Case</th>
<th>( c )</th>
<th>( n_f )</th>
<th>( n_g )</th>
<th>( n_h )</th>
<th>( n_{hid} )</th>
<th>( n_P )</th>
<th>( n_B )</th>
<th>( n_b )</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a)</td>
<td>/</td>
<td>1641</td>
<td>4</td>
<td>2590</td>
<td>0</td>
<td>0</td>
<td>801</td>
<td>8</td>
</tr>
<tr>
<td>(b)</td>
<td>10</td>
<td>1063</td>
<td>2</td>
<td>2595</td>
<td>0</td>
<td>0</td>
<td>801</td>
<td>8</td>
</tr>
<tr>
<td>(c)</td>
<td>10</td>
<td>876</td>
<td>2</td>
<td>2581</td>
<td>0</td>
<td>178</td>
<td>618</td>
<td>8</td>
</tr>
<tr>
<td>(d)</td>
<td>10</td>
<td>793</td>
<td>2</td>
<td>1878</td>
<td>36</td>
<td>18</td>
<td>530</td>
<td>2</td>
</tr>
</tbody>
</table>
and $\epsilon_1 = 0$. This means that when A.1 or A.2 terminates, $X^* \subseteq \bigcup_{y \in L} y$, where $X^*$ is the solution set of P.1 with
\[ S = \{ x \in X | g_i(x) \leq 0, i = 1, \ldots, n_g, \land h_j(x) \in [-\epsilon_E, \epsilon_E], j = 1, \ldots, n_h \}. \]
This estimate of $X^*_i$ is at least as satisfactory as estimates which are obtained using noninterval methods, since it is almost impossible to determine whether equality constraints are satisfied exactly in practice. Using a Gauss–Seidel sweep often produces a large reduction in computational cost for A.2, especially when $\epsilon_p$ is comparable with $\epsilon_c$.

Acknowledgements

The author is grateful to the referees for suggesting changes to the presentation of the material in this paper which greatly improved its clarity.

References