INTERVAL NEWTON / GENERALIZED BISECTION WHEN THERE ARE SINGULARITIES NEAR ROOTS

R. Baker KEARFOTT

Department of Mathematics, University of Southwestern Louisiana, Lafayette, Louisiana 70505, USA

Abstract

Interval Newton methods in conjunction with generalized bisection are important elements of algorithms which find the *global optimum* within a specified box $\mathbf{X} \subset \mathbb{R}^n$ of an objective function ϕ whose critical points are solutions to the system of nonlinear equations $F(X) = 0$ *with mathematical certainty*, even in finite precision arithmetic. The overall efficiency of such a scheme depends on the power of the interval Newton method to reduce the widths of the coordinate intervals of the box. Thus, though the generalized bisection method will still converge in a box which contains a critical point at which the Jacobian matrix is singular, the process is much more costly in that case. Here, we propose modifications which make the generalized bisection method isolate singular solutions more efficiently. These modifications are based on an observation about the verification property of interval Newton methods and on techniques for detecting the singularity and removing the region containing it. The modifications assume no special structure for F. Additionally, one of the observations should also make the algorithm more efficient when finding nonsingular solutions. We present results of computational experiments.

Keywords: Nonlinear algebraic systems, Newton's method, interval arithmetic, Gauss–Seidel method, global optimization, singularities.

1. Motivation, introduction, and notation

The general problem we address is:

Find, with certainty, the global optimum of the nonlinear objective function

$$\phi(X) = \phi(x_1, x_2, \ldots, x_n), \tag{1.1(a)}$$

where bounds $\underline{x}_i$ and $\overline{x}_i$ are known such that

$$\underline{x}_i \leqslant x_i \leqslant \overline{x}_i \text{ for } 1 \leqslant i \leqslant n.$$

A successful approach to this problem is generalized bisection in conjunction with interval Newton methods. The interval Newton method enables us to determine critical points (that is, roots of the gradient F of ϕ), whereas various techniques enable us to eliminate regions containing critical points which do not correspond to the *global* optimum before excessive effort is spent finding them. Thus, a related but more difficult problem is

Find, with certainty, approximations to all solutions of the nonlinear system

$$F(X) = (f_1(x_1, x_2, \ldots, x_n), \ldots, f_n(x_1, x_2, \ldots, x_n)) = 0, \qquad (1.1(\text{b}))$$

where bounds $\underline{x}_i$ and $\overline{x}_i$ are known such that

$$\underline{x}_i \leqslant x_i \leqslant \overline{x}_i \text{ for } 1 \leqslant i \leqslant n.$$

We write $X = (x_1, x_2, \ldots, x_n)$, and we denote the box given by the inequalities on the variables x_i by $\mathbf{B}$ [1].

Interval Newton algorithms for solving (1.1(a)) can be thought of as straightforward modifications of algorithms for solving (1.1(b)); see, the comprehensive reference [17] or, for example, [3] for a description of techniques particular to (1.1(a)). However, techniques for solving (1.1(b)) efficiently will similarly increase, in general, the efficiency of solution of (1.1(a)). For this reason, we concentrate on (1.1(b)) in the remainder of this paper. Along these lines, we will refer to the *Jacobian matrix* of F instead of the *Hessian matrix* of ϕ.

Interval Newton/generalized bisection methods for (1.1(b)) are described in [4–6,12,13,15,18] etc. For an introduction to the interval arithmetic underlying these methods, see [1,11], the recent review [9], etc. Also, the book [14] contains an overview of interval methods for linear and nonlinear systems of equations.

In these methods, we first transform $F(X) = 0$ to the linear interval system

$$\mathbf{F}'(\mathbf{X}_k)(\tilde{\mathbf{X}}_k - X_k) = -F(X_k), \qquad (1.2)$$

where $\mathbf{F}'(\mathbf{X}_k)$ is a suitable (such as an elementwise) interval extension [2] of the Jacobian matrix over the box $\mathbf{X}_k$ (with $\mathbf{X}_0 = \mathbf{B}$), and where $X_k \in \mathbf{X}_k$ represents a predictor or initial guess point. We note that (1.2) must be understood not as an ordinary equation but as the set of all linear systems of equations $A(X - X_k) = -F(X_k)$ as A ranges over all matrices which are contained in the interval matrix $\mathbf{F}'(\mathbf{X}_k)$. If we then formally solve (1.2) using interval arithmetic, the resulting box $\tilde{\mathbf{X}}_k$, which actually just satisfies

$$\mathbf{F}'(\mathbf{X}_k)(\tilde{\mathbf{X}}_k - X_k) \supset -F(X_k), \qquad (1.2(\text{b}))$$

will contain all solutions to all such systems $A(X - X_k) = -F(X_k)$. Furthermore, if each row of $\mathbf{F}'$ contains all possible vector values that the corresponding row of the scalar Jacobian matrix $F'(X)$ takes on as X ranges over all vectors in $\mathbf{X}_k$, then it follows from the mean value theorem that $\mathbf{X}_k$ will contain all solutions to $F(X) = 0$. We then define the next iterate $\mathbf{X}_{k+1}$ by

$$\mathbf{X}_{k+1} = \mathbf{X}_k \cap \tilde{\mathbf{X}}_k. \qquad (1.3)$$

This scheme is termed an *interval Newton method*.

[1] Throughout the paper, we will denote interval quantities with boldface letters. Vectors will be denoted with capital letters.

[2] Interval extensions of a function may be defined by simply evaluating the functions in interval arithmetic. The result of such a computation is an interval which contains the range of the function over the interval argument. Consult the introductions in [1], [11], and the recent review [9]; consult [16] for an in-depth treatment of interval methods for the range of functions.

If the coordinate intervals of $\mathbf{X}_{k+1}$ are not smaller than those of $\mathbf{X}_k$, then we may bisect one of these intervals to form two new boxes; we then continue the iteration with one of these boxes, and put the other one on a stack for later consideration. As explained in [5,6,12], and elsewhere, the following fact (from [13], p. 263) allows such a composite generalized bisection algorithm to compute all solutions to (1.1(b)) *with mathematical certainty*. For many methods of solving (1.2),

> if $\tilde{\mathbf{X}}_k \subseteq \mathbf{X}_k$, then the system of equations in (1.1) has a unique solution in $\mathbf{X}_k$. Conversely, if $\tilde{\mathbf{X}}_k \cap \mathbf{X}_k = \varnothing$ then there are no solutions of the system in (1.1) in $\mathbf{X}_k$. (1.4)

We now present a simplified version of the generalized bisection algorithm in [6]. [3]

ALGORITHM 1.1
Basic generalized bisection algorithm

```
1. (Initialization phase)
   (a) Input a tolerance $\epsilon$ such that no box will have a coordinate width less than $\epsilon$.
   (b) Input a tolerance $\epsilon_F$ such that we do no further computations on an $\mathbf{X}$ if $\| F(X) \|_\infty < \epsilon_F$ for $X \in \mathbf{X}$.
   (c) $\mathbf{X}_k \leftarrow \mathbf{B}$.
2. (Bisection)
   (a) If $\mathbf{X}_k = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n)$, where $\mathbf{x}_j = [\underline{x}_j, \overline{x}_j]$, then choose a coordinate $i$ in which to bisect.
   (b) Form two new boxes $\mathbf{X}_k^1$ and $\mathbf{X}_k^2$ by replacing $\mathbf{x}_i$ in $\mathbf{X}_k$ by either $[\omega_i, \overline{x}_i]$ or $[\underline{x}_i, \omega_i]$, where $\omega_i = (\underline{x}_i + \overline{x}_i)/2$.
   (c) Place either $\mathbf{X}_k^1$ or $\mathbf{X}_k^2$ on a stack $\mathscr{S}$ for later consideration, and replace $\mathbf{X}_k$ with the other one.
3. (Interval Newton method and root storage)
   (a) (Test for convergence)
       (i) If the width of at least one coordinate $\mathbf{x}_j$ of $\mathbf{X}_k$ is greater than $\epsilon$, then compute the interval vector $\mathbf{F}(\mathbf{X})$ for use in (ii) below.
       (ii) If the width of each coordinate $\mathbf{x}_j$ of $\mathbf{X}_k$ is less than $\epsilon$, or if $\| F(X) \|_\infty < \epsilon_F$ then
           ($\alpha$) Store $\mathbf{X}_k$ in a list $\mathscr{L}'$ of small boxes which possibly contain roots.
           ($\beta$) If the stack $\mathscr{S}$ is empty, then stop. Otherwise, pop a box from $\mathscr{S}$, let that box become $\mathbf{X}_k$, and return to the beginning of this step.
```

[3] For clarity, we do not include the "expansion step", which is step 4 of algorithm 3.1 in [6], although the implementation in the experiments in this paper has it. In our experience, this step usually does not affect the various measures of efficiency for the algorithm.

(b) (Obtain the function and Jacobian for (1.2).)
    (i) Compute the interval Jacobian matrix $\mathbf{F}'(\mathbf{X}_k)$.
    (ii) Compute $F(X_k)$, using interval arithmetic to bound the roundoff error.
(c) (Bound the solution set in (1.2).) Use some method to compute an interval enclosure $\tilde{\mathbf{X}}_k$ to the solution set of the interval linear system (1.1(b)).
(d) If $\tilde{\mathbf{X}}_k \subseteq \mathbf{X}_k$, then do the following.
    (i) Store $\mathbf{X}_k$ in a list $\mathscr{L}$ of boxes which contain unique roots.
    (ii) If the stack $\mathscr{S}$ is empty, then stop. Otherwise, pop a box from $\mathscr{S}$, let that box become $\mathbf{X}_k$, and return to the beginning of step 3(a).
(e) If $\tilde{\mathbf{X}}_k \cap \mathbf{X}_k$ is sufficiently smaller than $\mathbf{X}_k$, then replace $\mathbf{X}_k$ by $\tilde{\mathbf{X}}_k \cap \mathbf{X}_k$ and return to step 3(a). Otherwise, replace $\mathbf{X}_k$ by $\tilde{\mathbf{X}}_k \cap \mathbf{X}_k$ and return to step 2.
(f) If $\tilde{\mathbf{X}}_k \cap \mathbf{X}_k = \varnothing$ then stop if the stack $\mathscr{S}$ is empty; otherwise, pop a box from $\mathscr{S}$, let that box become $\mathbf{X}_k$, and return to the beginning of step (3a).

In step 3(e), we may say $\tilde{\mathbf{X}}_k$ is sufficiently smaller than $\mathbf{X}_k$ if there is a j such that $\bar{x}_j - \underline{x}_j > \epsilon$ and $(\bar{\tilde{x}} - \underline{\tilde{x}}_j) \leqslant (\bar{x} - \underline{x}_j)/2$. Such a condition will ensure the overall convergence of the algorithm, since it guarantees that each step, whether interval Gauss–Seidel or bisection, will reduce one of the coordinates by at least a factor of 2. In practice, however, we have found a volume ratio (which, appropriately implemented, also implies convergence) to be effective; we continue to use the Gauss–Seidel iteration after a sweep of all n coordinates only if

$$\prod_{\substack{i=1\\ w(\mathbf{x}_i)>\epsilon}}^{n} w(\mathbf{x}_i^{+}) < \eta \prod_{\substack{i=1\\ w(x_i)>\epsilon}}^{n} w(\mathbf{x}_i), \tag{1.5}$$

for some η with $0 < \eta < 1$ ($\eta = 0.6$ works well), where $\mathbf{x}_i^{+} = \tilde{\mathbf{x}}_i \cap \mathbf{x}_i$.

We use the following notation. We write $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n)$ for $\mathbf{X}_k$ and we write $\mathbf{A}_{i,j}$ for the interval in the ith row and jth column of $\mathbf{A} = \mathbf{F}'(\mathbf{X})$. Similarly, we write[4] $F(X_k) = F = (\mathbf{f}_1, \mathbf{f}_2, \ldots, \mathbf{f}_n)$, and $X_k = (x_1, x_2, \ldots, x_n)$, so that (1.2) becomes

$$\mathbf{A}\big(\tilde{\mathbf{X}}_k - X_k\big) = -F \tag{1.6}$$

We generally precondition (1.6); i.e., we multiply by a matrix Y to obtain

$$Y\mathbf{A}\big(\tilde{\mathbf{X}}_k - X_k\big) = -YF. \tag{1.7}$$

Let $Y_i = (y_1, y_2, \ldots, y_n)$ denote the ith row of the preconditioner, let

$$\mathbf{k}_i = Y_i F,$$

and let

$$Y_i\mathbf{A} = \mathbf{G}_i = (\mathbf{G}_{i,1}, \mathbf{G}_{i,2}, \ldots, \mathbf{G}_{i,n}) = \left(\left[\underline{g}_{i,1}, \bar{g}_{i,1}\right], \left[\underline{g}_{i,2}, \bar{g}_{i,2}\right], \ldots, \left[\underline{g}_{i,n}, \bar{g}_{i,n}\right]\right).$$

[4] We denote the components of F as boldface intervals, since they must be evaluated in interval arithmetic with directed roundings or else roundoff error may cause algorithm 1.1 to miss a root.

With the above notation, we have the following version of the interval Newton method for step 3(c) of algorithm 1.1 which usually works well.

```
ALGORITHM 1.2
Preconditioned version of interval Gauss–Seidel; see [8]
Do the following for $i = 1$ to $n$.
1. (Update a coordinate.)
   (a) Compute the preconditioner row $Y_i$ as the linear programming preconditioner described in [8].
   (b) Compute $\mathbf{k}_i$ and $\mathbf{G}_i$.
   (c) Compute
       $$\tilde{\mathbf{x}}_i = x_i - \left[\mathbf{k}_i + \sum_{\substack{j=1\\ j\neq i}}^{n} \mathbf{G}_{i,j}(\mathbf{x}_j - x_j)\right] / \mathbf{G}_{i,i} \qquad (1.8)$$
       using interval arithmetic.
2. (The new box is empty.) If $\tilde{\mathbf{x}}_i \cap \mathbf{x}_i = \varnothing$, then signal that there is no root of $F$ in $\mathbf{X}$, and continue the generalized bisection algorithm.
3. (The new box is non-empty; prepare for the next coordinate.)
   (a) Replace $\mathbf{x}_i$ by $\mathbf{x}_i \cap \tilde{\mathbf{x}}_i$.
   (b) Possibly re-evaluate $\mathbf{F}'(\mathbf{X}_k)$ to replace $\mathbf{A}$ by an interval matrix whose corresponding widths are smaller.
```

In step 2(a) of algorithm 1.1, we have found *maximal smear* to be appropriate for determining which coordinate to bisect. That is, we bisect the coordinate direction j for which σ_j is maximum, where

$$\sigma_j = \max_{1\leq i\leq n} \left\{ |\underline{a}_{i,j}|, |\overline{a}_{i,j}| \right\} (\overline{x}_j - \underline{x}_j). \qquad (1.9)$$

This coordinate direction is, roughly, the one in which the values of the f_i change most rapidly relative to the individual widths of the present box $\mathbf{X}$.

Algorithm 1.1 must eventually complete with (possibly empty) lists of boxes $\mathscr{L}$ and $\mathscr{L}'$, such that all roots of F in $\mathbf{B}$ are contained in boxes in $\mathscr{L}$ or $\mathscr{L}'$, and each box in $\mathscr{L}$ contains a unique root; compare with the convergence analysis in [5]. However, the cost bound in [5] is very pessimistic, and the actual efficiency of algorithm 1.1 depends on how well algorithm 1.2 (step 3(c) of algorithm 1.1) finds the solution bounds $\tilde{\mathbf{X}}_k$ in (1.2). If the widths of the components of $\mathbf{X}$ are sufficiently small, if the Jacobian matrix is reasonably well-conditioned, and if the interval extension $\mathbf{F}'$ to the Jacobian matrix gives reasonably sharp bounds on the range of the Jacobian matrix, then the widths of the components of $\tilde{\mathbf{X}}_k$ are smaller than those of $\mathbf{X}$, and iteration of algorithm 1.2 will reduce them further, until the condition in step 3(d) holds. (In fact, this interval Newton method is locally quadratically convergent in the sense that the widths go to zero at that rate.) If not, then bisection reduces the size of $\tilde{\mathbf{X}}_k$ slowly, especially when the dimension n is large, and many more boxes must be considered.

In this paper, we present techniques which are useful when algorithm 1.2 does not give bounds $\tilde{\mathbf{X}}_k$ with smaller widths, because the Jacobian matrix is ill-conditioned or singular near the root. These techniques do not assume anything about the structure of the singularity, and they may be embedded into algorithm 1.1 without adversely affecting its efficiency on non-singular problems.

The techniques are based on

(i) an observation concerning a componentwise variant of (1.4); and

(ii) an algorithmic technique and set of heuristics for astutely "trisecting" a box which contains a singular root.

We formally present the componentwise variant of (1.4) in section 2. In section 3, we give the trisection algorithm, the heuristics, and a corresponding modified version of algorithm 1.1. In section 4, we present computational results from the test set in [6] and additional functions with singular roots. We summarize in section 5.

2. A componentwise root inclusion test

Suppose $X_* = (r_1, r_2, \ldots, r_n)$ is a root ($F(X_*) = 0$), such that the Jacobian matrix $F'(X_*)$ is singular, and suppose $X_* \in \mathbf{X}_k$. Then it is impossible for $\tilde{\mathbf{X}}_k \subseteq \mathbf{X}_k$ as in (1.4). (To see this, note that $\mathbf{F}'(\mathbf{X}_k)$ contains a singular matrix, so that $\tilde{\mathbf{X}}_k$ cannot be bounded.) [5] However, if we choose the preconditioner Y_i as in algorithm 1.2, then in many cases each width $w(\tilde{\mathbf{x}}_i)$ is minimal, given the interval extension $\mathbf{F}'(\mathbf{X}_k)$. Thus, even if $\mathbf{X}_k$ contains a point X_* with $F'(X_*)$ singular, we often have $\tilde{\mathbf{x}}_i \subseteq \mathbf{x}_i$ for some (but not all) i.

The above considerations lead us to examine weaker forms of (1.4) for singular systems. We obtained some preliminary results in [7], in which we singled out certain directions in which F was singular, and examined a related non-singular subproblem in a lower-dimensional space. In fact, by viewing the problem slightly differently, we may do this more directly and efficiently within the framework of algorithm 1.1. The idea is to view the system $F(X) = 0$ as a lower-dimensional system which is parametrized in terms of the variables x_i for which $\tilde{\mathbf{x}}_i \not\subseteq \mathbf{x}_i$. We have

THEOREM 2.1

Let $\tilde{\mathbf{x}}_i$ be computed in algorithm 1.2, for each i with $1 \leqslant i \leqslant n$. Let

$$\mathcal{N}_{\text{conv}} = \{ i \mid \tilde{\mathbf{x}}_i \subseteq \mathbf{x}_i \} = \{ \iota_j \}_{j=1}^{n_c},$$

[5] Similarly, if there is a point $X \in \mathbf{X}_k$ at which $F'(X)$ is ill-conditioned, then we can expect at least one component interval of $\tilde{\mathbf{X}}_k$ to be large.

and let

$$\mathcal{N}_{\text{div}} = \{ i \mid \tilde{\mathbf{x}}_i \not\subseteq \mathbf{x}_i \} = \{ \mu_j \}_{j=1}^{n_d}.$$

If $X = (x_1, x_2, \ldots, x_n)$, then let $\check{F}: \mathbb{R}^{n_c} \to \mathbb{R}^{n_c}$ be given by

$$\check{F}\left(x_{\iota_1}, x_{\iota_2}, \ldots, x_{\iota_{n_c}} \mid x_{\mu_1}, x_{\mu_2}, \ldots, x_{\mu_{n_d}} \right) = \left(Y_{\iota_1} F(X), Y_{\iota_2} F(X), \ldots, Y_{\iota_{n_c}} F(X) \right).$$

Define

$$\mathbf{X}_{\text{conv}} = \left\{ \left(x_{\iota_1}, x_{\iota_2}, \ldots, x_{\iota_{n_c}} \right) \mid x_{\iota_j} \in \mathbf{x}_{\iota_j} \text{ for } \iota_j \in \mathcal{N}_{\text{conv}} \right\},$$

and

$$\mathbf{X}_{\text{div}} = \left\{ \left(x_{\mu_1}, x_{\mu_2}, \ldots, x_{\mu_{n_d}} \right) \mid x_{\mu_j} \in \mathbf{x}_{\mu_j} \text{ for } \mu_j \in \mathcal{N}_{\text{div}} \right\}.$$

Then, for each $(x_{\mu_1}, x_{\mu_2}, \ldots, x_{\mu_{n_d}}) \in \mathbf{X}_{\text{div}}$, $\check{F}(x_{\iota_1}, x_{\iota_2}, \ldots, x_{\iota_{n_c}}) = 0$ has a unique solution in $\mathbf{X}_{\text{conv}}$.

Theorem 2.1 states that, if the interval Gauss–Seidel method (algorithm 2.1) reduces the widths of n_c of the component intervals, then, for each choice for each of the remaining variables, there is a unique set of values of the corresponding variables within those intervals for which n_c linear combinations of the function, defined by the preconditioner rows, simultaneously equals zero.

Proof of theorem 2.1

The proof is similar to the proof of theorem 2.3 in [10]. In each case, the interval linear system analogous to (1.2) contains the solutions to a parametrized set of nonlinear systems, so that conclusions based on (1.4) apply to each element of the set.

Specifically, pick any particular $(x_{\mu_1}, x_{\mu_2}, \ldots, x_{\mu_{n_d}}) \in \mathbf{X}_{\text{div}}$. Then $\check{F}(x_{\iota_1}, x_{\iota_2}, \ldots, x_{\iota_{n_c}})$ is simply a function from $\mathbb{R}^{n_c}$ to $\mathbb{R}^{n_c}$ which obeys the hypotheses in the first part of (1.4). Furthermore, by the construction of $\mathbf{X}_{\text{conv}}$, $\tilde{\mathbf{X}}_{\text{conv}} \subseteq \mathbf{X}_{\text{conv}}$, so $\check{F}$ has a unique solution in $\mathbf{X}_{\text{conv}}$. $\square$

The fact that $\tilde{\mathbf{x}}_i \subseteq \mathbf{x}_x$ for one or more i is evidence that the box $\mathbf{X}_k$ is small enough for the linear interval system (1.2) to model the local behavior of F (in at least some components). Additionally, it allows us to reduce the widths of certain coordinates of $\mathbf{X}_k$ through the interval Newton method, thus avoiding the necessity to bisect those coordinates. This, in turn, usually results in less total operations to complete the generalized bisection algorithm. The following corollary to theorem 2.1 clarifies these facts.

COROLLARY 2.2

Suppose the box $\mathbf{X}_k$ has a non-empty index set $\mathcal{N}_{\text{conv}}$ associated with it, as in theorem 2.1. Suppose also that $\mathbf{X}$ is any box obtained from $\mathbf{X}_k$ by repeated application (in any order) of step 2(a) or step 3(e) within algorithm 1.1, but under the assumption that the coordinate index i in step 2(c) is always chosen from $\mathcal{N}_{\text{div}}$. Then the conclusion of theorem 2.1 is still true, with the same $\mathcal{N}_{\text{conv}}$ and $\mathcal{N}_{\text{div}}$, but with $\mathbf{X}$ replacing $\mathbf{X}_k$.

Proof

We may view algorithm 1.1 as producing a hierarchy of boxes: step 2 produces two boxes below $\mathbf{X}_k$, whereas step 3(e) produces one such box. The corollary follows by induction on the level of the boxes in this hierarchy. To this end, assume that, instead of being the initial box, $\mathbf{X}_k$ is an arbitrary box in the hierarchy, and that the conclusions of theorem 2.1 are true for $\mathbf{X}_k$. Then, if $\mathbf{X}_k$ passes through step 2, the boxes $\mathbf{X}_k^1$ and $\mathbf{X}_k^2$ have coordinate sets $\mathbf{X}_{\text{conv}}^1$ and $\mathbf{X}_{\text{conv}}^2$ which are identical to the set $\mathbf{X}_{\text{conv}}$ corresponding to $\mathbf{X}_k$, but have coordinate sets $\mathbf{X}_{\text{div}}^1$ and $\mathbf{X}_{\text{div}}^2$ which are strict subsets of those of $\mathbf{X}_k$. Since the conclusion of theorem 2.1 held for *any* $(x_{\mu_1}, x_{\mu_2}, \ldots, x_{\mu_{n_d}}) \in \mathbf{X}_{\text{div}}$, it must hold when $\mathbf{X}_{\text{div}}$ is replaced by either $\mathbf{X}_{\text{div}}^1$ or $\mathbf{X}_{\text{div}}^2$.

Now suppose that $\mathbf{X}_k$ passes through step 3(e) of algorithm 1.1. Then a single new box $\mathbf{X}^+ = \mathbf{X}_k \cap \tilde{\mathbf{X}}_k$ is produced, whose coordinates can be grouped into $\tilde{\mathbf{X}}_{\text{conv}}^+ = \tilde{\mathbf{X}}_{\text{conv}} \cap \mathbf{X}_{\text{conv}}$ and $\tilde{\mathbf{X}}_{\text{div}}^+ = \tilde{\mathbf{X}}_{\text{div}} \cap \mathbf{X}_{\text{div}}$. But fix any particular $(x_{\mu_1}, x_{\mu_2}, \ldots, x_{\mu_{n_d}}) \in \mathbf{X}_{\text{div}}$, and apply the interval Newton method with the point $(x_{\mu_1}, x_{\mu_2}, \ldots, x_{\mu_{n_d}}) \in \tilde{\mathbf{X}}_{\text{div}}^+$ replacing $\mathbf{X}_{\text{div}}$. Then (from monotone inclusion properties of interval arithmetic), the set which would be stored in step 3(e) must be contained in the degenerate box with coordinate intervals taken from $\tilde{\mathbf{X}}_{\text{conv}}^+$ and $(x_{\mu_1}, x_{\mu_2}, \ldots, x_{\mu_{n_d}})$. However, any solutions of $\check{F} = 0$ (with parameters $(x_{\mu_1}, x_{\mu_2}, \ldots, x_{\mu_{n_d}})$) in $\mathbf{X}_{\text{conv}}$ must also be in $\tilde{\mathbf{X}}_{\text{conv}}^+$. Therefore, for each $(x_{\mu_1}, x_{\mu_2}, \ldots, x_{\mu_{n_d}}) \in \tilde{\mathbf{X}}_{\text{div}}^+$, there is a unique solution of $\check{F} = 0$ in $\tilde{\mathbf{X}}_{\text{conv}}^+$. This concludes the proof of the corollary. □

We note that, despite corollary 2.2, boxes may be produced from $\mathbf{X}_k$ which do not contain any roots of the *full* function F. Such boxes are detected in step 3(a) by observation that the interval function value does not contain zero. They also may be detected in step 3(e) if $\tilde{\mathbf{X}}_k \cap \mathbf{X}_k = \varnothing$.

Alternately, we may conclude that there is a unique solution of $F(X) = 0$ in one of the boxes. The following theorem tells us that we need only check $\tilde{\mathbf{X}}_{\text{div}} \subseteq \mathbf{X}_{\text{div}}$.

THEOREM 2.3

Suppose $\mathbf{X}$ is a box produced from a box $\mathbf{X}_k$ for which the hypotheses of theorem 2.1 hold with coordinate bound lists $\mathbf{X}_{\text{conv}}^0$ and $\mathbf{X}_{\text{div}}^0$, where we assume, as in corollary 2.2, that only coordinate directions represented in $\mathcal{N}_{\text{div}}$ were bisected. Let the corresponding coordinate bound lists for $\mathbf{X}$ be $\mathbf{X}_{\text{conv}}$ and $\mathbf{X}_{\text{div}}$. Suppose that $\mathbf{X}$ enters steps 3(c) and 3(d) of algorithm 1.1; let the image box $\check{\mathbf{X}}$

have coordinate bound lists $\tilde{\mathbf{X}}_{\text{conv}}$ and $\tilde{\mathbf{X}}_{\text{div}}$. If $\tilde{\mathbf{X}}_{\text{div}} \subseteq \mathbf{X}_{\text{div}}$, then $F(X)=0$ has a unique solution within X.

Proof

First, suppose that we apply algorithm 1.2 to any box $\mathbf{X}_{\text{between}}$ whose ith coordinate interval is the ith coordinate of the image $\tilde{\mathbf{X}}_k$ of $\mathbf{X}_k$ if $i \in \mathcal{N}_{\text{conv}}$, and whose ith coordinate interval is contained in the ith coordinate interval of $\tilde{\mathbf{X}}_k$ if $i \in \mathcal{N}_{\text{div}}$. Inclusion monotonicity then implies that, if we use the appropriate preconditioner row, the ith coordinate of the image $\tilde{\mathbf{X}}_{\text{between}}$ is contained in the ith coordinate of $\mathbf{X}_{\text{between}}$ for $i \in \mathcal{N}_{\text{conv}}$. If we then apply mathematical induction (and, without loss of generality, assume use of the appropriate preconditioner rows), we may conclude that the ith coordinate of the image $\tilde{\mathbf{X}}_{\text{between}}$ is contained in the ith coordinate of $\mathbf{X}_{\text{between}}$ for $i \in \mathcal{N}_{\text{conv}}$, if $\mathbf{X}_{\text{between}}$ is *any* box produced from $\mathbf{X}_k$ as in corollary 2.2. Thus, $\tilde{\mathbf{X}} \subseteq \mathbf{X}$. □

Theorem 2.3 is of particular use in practice, since repetition of step 3(c) in algorithm 1.1 typically causes convergence of one or more coordinate intervals, so that strict containment of the image intervals in subsequent boxes cannot be expected. In those cases, the outward rounding process often precludes assertion of containment. With theorem 2.3, we generally need not check containment once such convergence has occurred.

3. Trisection and other algorithms

In this section, we first introduce a process for algorithmically handling roots at which the Jacobian matrix is singular or ill-conditioned. We then present a modified version of algorithm 1.1 which will be more efficient at isolating roots at which the Jacobian matrix is ill-conditioned.

The algorithm for singularities is based on "trisection" of the box, and is applied after step 3(d) of algorithm 1.1. The algorithm incorporates a heuristic for determining when the interval linear system (1.2) adequately models the original nonlinear system $F(X)=0$; namely, it tests for singularity provided the set $\mathcal{N}_{\text{conv}}$ in theorem 2.1 is nonempty.

ALGORITHM 3.1

Algorithmic removal of singular roots.

Let $\mathbf{X}_k$ be the current box before step 3(e) of algorithm 1.1, and let X_k be the corresponding guess point for (1.2). Let $\mathcal{N}_{\text{conv}}$ and $\mathcal{N}_{\text{div}}$ as in theorem 2.1 be given; also maintain a list of those coordinates $\mathcal{N}_{\text{already}}$ of $\mathbf{X}_k$ which have been previously produced from the trisection process in step 3 of this algorithm. Then do the following if $\mathcal{N}_{\text{conv}} \neq \varnothing$.

1. Apply the classical Newton's method, with starting point X_k, to find a root of $F(X)=0$, using tolerances scaled appropriately for the linear convergence near singular roots.
   - If Newton's method does not converge to a point within the box, then return.
   - Otherwise, do the remaining steps of this algorithm.
2. Let $X_* \in \mathbf{X}_k$ be the solution to which Newton's method has converged. Compute the condition number (or estimate thereof) for the Jacobian matrix $F'(X_*)$.
   - If the condition number is less than a prescribed tolerance $\kappa_{\max}$, then return.
   - Otherwise, continue to step 3.
3. (Actual trisection)
   (a) Find t such that
   $$\sigma_t = \max_{\substack{1 \leqslant j \leqslant n \\ j \notin \mathcal{N}_{\text{conv}} \\ j \notin \mathcal{N}_{\text{already}}}} \sigma_j,$$
   where
   $$\sigma_j = \max_{1 \leqslant i \leqslant n} \{ |\underline{a}_{i,j}|, |\overline{a}_{i,j}| \} (\overline{x}_j - \underline{x}_j).$$
   (b) Given a domain tolerance $\epsilon_{\max}$ for the width of a coordinate at a singular solution, form (one, two, or) three new boxes $\mathbf{X}_k^1$, $\mathbf{X}_k^2$, and $\mathbf{X}_k^3$, such that
   (i) $\mathbf{X}_k^1$ is obtained from $\mathbf{X}_k$ by replacing the tth coordinate interval $[\underline{x}_t^{(k)}, \overline{x}_t^{(k)}]$ of $\mathbf{X}_k$ by $[\underline{x}_t^{(k)}, x_t^{(*)} - \epsilon_{\max}\underline{x}_t^{(k)}]$, provided $x_t^{(*)} - \epsilon_{\max}\underline{x}_t^{(k)} \geqslant \underline{x}_t^{(k)}$;
   (ii) $\mathbf{X}_k^2$ is obtained from $\mathbf{X}_k$ by replacing the tth coordinate interval $[\underline{x}_t^{(k)}, \overline{x}_t^{(k)}]$ of $\mathbf{X}_k$ by $[x_t^{(*)} + \epsilon_{\max}\overline{x}_t^{(k)}, \overline{x}_t^{(k)}]$, provided $x_t^{(*)} + \epsilon_{\max}\overline{x}_t^{(k)} \leqslant \overline{x}_t^{(k)}$;
   (ii) $\mathbf{X}_k^3$ is obtained from $\mathbf{X}_k$ by replacing the tth coordinate interval $[\underline{x}_t^{(k)}, \overline{x}_t^{(k)}]$ of $\mathbf{X}_k$ by $[x_t^{(*)} - \epsilon_{\max}\underline{x}_t^{(k)}, x_t^{(*)} + \epsilon_{\max}\overline{x}_t^{(k)}]$.
4. (Adjusting information for the main algorithm)
   (a) Put $\mathbf{X}_k^1$ and $\mathbf{X}_k^2$ on the stack of boxes to be considered later.
   (b) Replace the current box $\mathbf{X}_k$ by $\mathbf{X}_k^3$.
   (c) Place t in the set $\mathcal{N}_{\text{already}}$ corresponding to the current box $\mathbf{X}_k^3$ (but not $\mathbf{X}_k^1$ or $\mathbf{X}_k^2$).

Figure 1 illustrates execution of step 3(b) of algorithm 3.1.

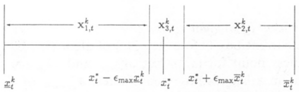

Fig. 1. Illustration of the three subintervals into which the tth coordinate is divided in step 3(b) of algorithm 3.1.

To incorporate algorithm 3.1, we modify the basic generalized bisection algorithm. For example, we do not wish to bisect a coordinate of a box which was produced from some $\mathbf{X}_k^3$ and whose index is in $\mathcal{N}_{\text{already}}$. Likewise, it is unnecessary to bisect in coordinate directions whose indices are in $\mathcal{N}_{\text{conv}}$. Furthermore, we should not measure the widths of coordinate intervals whose indices are in $\mathcal{N}_{\text{conv}}$ or $\mathcal{N}_{\text{already}}$ when testing the size of the box for further bisection. Finally, we gain efficiency if we take account of theorem 2.3 when determining the inclusion in (1.4) or in step 3(d) of algorithm 1.1. The following modified generalized bisection algorithm takes account of these considerations.

ALGORITHM 3.2

Generalized bisection with trisection to handle singularities

1. (Initialization phase)
   - (a) Input a tolerance ϵ such that no box will have a coordinate width less than ϵ.
   - (b) Input a tolerance ϵ_F such that we do no further computations on an $\mathbf{X}$ if $\| F(X) \|_\infty < \epsilon_F$ for $X \in \mathbf{X}$.
   - (c) $\mathbf{X}_k \leftarrow \mathbf{B}$.
   - (d) $\mathcal{N}_{\text{conv}} \leftarrow \varnothing$ and $\mathcal{N}_{\text{already}} \leftarrow \varnothing$.
2. (Bisection) Do the following step only if
$$\mathcal{N}_{\text{conv}} \cup \mathcal{N}_{\text{already}} \neq \{1, 2, \ldots, n\}.$$
   - (a) If $\mathbf{X}_k = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n)$, where $\mathbf{x}_j = [\underline{x}_j, \overline{x}_j]$, then choose a coordinate $i \notin \mathcal{N}_{\text{conv}} \cup \mathcal{N}_{\text{already}}$ in which to bisect.
   - (b) Form two new boxes $\mathbf{X}_k^1$ and $\mathbf{X}_k^2$ by replacing $\mathbf{x}_i$ in $\mathbf{X}_k$ by either $[\omega_i, \overline{x}_i]$ or $[\underline{x}_i, \omega_i]$, where $\omega_i = (\underline{x}_i + \overline{x}_i)/2$.
   - (c) Place either $\mathbf{X}_k^1$ or $\mathbf{X}_k^2$ on a stack $\mathcal{S}$ for later consideration (along with $\mathcal{N}_{\text{conv}}$ and $\mathcal{N}_{\text{already}}$), and replace $\mathbf{X}_k$ with the other one.
3. (Interval Newton method and root storage)
   - (a) (Test for convergence)
     - (i) If the width of at least one coordinate $\mathbf{x}_j$ of $\mathbf{X}_k$ for $j \notin \mathcal{N}_{\text{conv}} \cup \mathcal{N}_{\text{already}}$ is greater than ϵ, then compute the interval vector $\mathbf{F}(\mathbf{X})$ for use in (ii) below.
     - (ii) If the width of each coordinate $\mathbf{x}_j$ of $\mathbf{X}_k$ with $j \notin \mathcal{N}_{\text{conv}} \cup \mathcal{N}_{\text{already}}$ is less than ϵ, or if $\| F(X) \|_\infty < \epsilon_F$ then
       - (α) Store $\mathbf{X}_k$ in a list $\mathcal{L}'$ of small boxes which possibly contain roots.
       - (β) If the stack $\mathcal{S}$ is empty, then stop. Otherwise, pop a box from $\mathcal{S}$ (along with $\mathcal{N}_{\text{conv}}$ and $\mathcal{N}_{\text{already}}$), let that box become $\mathbf{X}_k$, and return to the beginning of this step.
   - (b) (Obtain the function and Jacobian for (1.2).)
     - (i) Compute the interval Jacobian matrix $\mathbf{F}'(\mathbf{X}_k)$.
     - (ii) Compute $F(X_k)$, using interval arithmetic to bound the roundoff error.

(c) (Bound the solution set in (1.2).) Use algorithm 1.2 to compute an interval enclosure $\tilde{\mathbf{X}}_k$ to the solution set of the interval linear system (1.1(b)).
(d) If $\mathbf{x}_i \subseteq \tilde{\mathbf{x}}_i$ for each i with $i \notin \mathcal{N}_{\text{conv}}$, then do the following.
  (i) Store $\mathbf{X}_k$ in a list $\mathcal{L}$ of boxes which contain unique roots.
  (ii) If the stack $\mathcal{S}$ is empty, then stop. Otherwise, pop a box from $\mathcal{S}$ (along with $\mathcal{N}_{\text{conv}}$ and $\mathcal{N}_{\text{already}}$), let that box become $\mathbf{X}_k$, and return to the beginning of step 3(a).
(e) Update $\mathcal{N}_{\text{conv}}$ by taking the union of the old $\mathcal{N}_{\text{conv}}$ with those indices i from step 3(d) for which $\mathbf{x}_i \subseteq \tilde{\mathbf{x}}_i$.
(f) If $\mathcal{N}_{\text{conv}} \neq \varnothing$, then execute algorithm 3.1.
(g) If (1.6) holds, then replace $\mathbf{X}_k$ by $\tilde{\mathbf{X}}_k \cap \mathbf{X}_k$ and return to step 3(a). Otherwise, replace $\mathbf{X}_k$ by $\tilde{\mathbf{X}}_k \cap \mathbf{X}_k$ and return to step 2.
(h) If $\tilde{\mathbf{X}}_k \cap \mathbf{X}_k = \varnothing$ then stop if the stack $\mathcal{S}$ is empty; otherwise, pop a box from $\mathcal{S}$ (along with $\mathcal{N}_{\text{conv}}$ and $\mathcal{N}_{\text{already}}$), let that box become $\mathbf{X}_k$, and return to the beginning of step 3(a).

4. Numerical results

In this section, we report test results comparing algorithm 3.2 to the algorithm in [8], where we report on the basic generalized bisection algorithm in conjunction with the specially preconditioned interval Gauss–Seidel method (algorithm 1.2).

The test set is that from [8], which is that from [6]. Common tolerances and algorithmic details, with some minor modifications, are as in [8]. In trisection (algorithm 3.1) we took the range tolerance for the point Newton method in step 1 to be 10^{-12} and the domain tolerance to be $10^{-5} \| X_k \|_2$. In step 2 of algorithm 3.1, we took $\kappa_{\max} = 10^4$, and we took the minimum coordinate width in step 3(b) to be $\epsilon_{\max} = 10^{-3}$.

As in [8], table 1 gives estimates for the amount of work for each of the three methods. The first column gives the problem number as in [6], the second column gives the dimension n of the problem, and the third column gives the method, where "lp" refers to the basic method with the linear programming preconditioner, as in [8], and where "tri." refers to the trisection method (with algorithm 3.1 and algorithm 3.2) for singularities. Column 4 (NBOX) gives the total numbers of boxes considered in algorithm 1.1 or algorithm 3.2 (i.e. the number of times that step 3(c) is entered), column 5 (NFUN) gives the total number of interval function evaluations, column 6 (NJAC) gives the total number of interval Jacobian evaluations, and column 7 gives an estimate W_e for the total amount of work, which is computed as

$$W_e = \text{NFUN} + n\text{NJAC}.$$

We notice that algorithm 3.1 and algorithm 3.2 were markedly superior to the basic algorithm on problem 3, which has a rank-two defect in the Jacobian matrix

Table 1
Cost measures with and without trisection

#	n	meth.	NBOX	NFUN	NJAC	Est. work
1	2	tri.	41	73	32	137
		lp	41	69	28	115
2	2	tri.	53	86	33	152
		lp	53	86	33	152
3	4	tri.	25	40	13	92
		lp	480	726	246	1710
4	5	tri.	68	124	56	404
		lp	68	124	56	404
9	2	tri.	25	48	23	94
		lp	23	44	21	86
10	4	tri.	398	635	235	1575
		lp	771	1214	443	2986
11	8	tri.	571	973	321	3541
		lp	671	1141	371	4109
12	3	tri.	563	922	359	1999
		lp	551	914	363	2003
14	2	tri.	44	72	28	128
		lp	41	68	27	122
15	2	tri.	3	5	2	9
		lp	3	5	2	9
16	4	tri.	3	6	3	18
		lp	3	6	3	18
17	5	tri.	67	109	42	319
		lp	67	108	41	313
Tot.		tri.	1861	3093	1147	8468
		lp	2772	4505	1634	12024

at the root. Furthermore, using the new algorithm did not seem to unduly degrade the performance on the other problems; in fact, it seems to also work better when there are scaling difficulties, as in problem 10.

In addition to the efficiency improvements evident in table 1, algorithm 3.1 and algorithm 3.2 have certain qualitative advantages. For example, in problem 3 (a variant of Powell's singular function), only a single possible root-containing box was placed in the list $\mathscr{L}'$ (and none were deleted from $\mathscr{L}'$ in the deletion steps explained in [5] and [6]), whereas, when the basic algorithm was used, 5 boxes were placed in $\mathscr{L}'$ (giving a redundant listing of the root), and 3 boxes

Table 2
Total clock times for the two algorithms

#	tri.	lp
1	0.07	0.06
2	0.13	0.18
3	0.17	1.63
4	1.65	1.56
9	0.07	0.05
10	4.61	6.90
11	23.22	16.00
12	2.73	2.21
14	0.06	0.06
15	0.004	0.004
16	0.03	0.03
17	1.43	1.26
All	34.22	29.94

were deleted from $\mathscr{L}'$. Also, theorem 3.2 came into play effectively in problem 11, which has 16 distinct roots, none of which corresponded to a singular Jacobian matrix. In the basic algorithm (with the linear programming preconditioner), the 16 root-containing boxes were placed in the list $\mathscr{L}'$, for which there is no definite affirmation that unique roots have been isolated; however, with algorithm 3.1 and algorithm 3.2, all 16 boxes were placed in the list $\mathscr{L}$, for which there is verification that each box contains a unique root.

Total execution times for the algorithms can only be taken as another relative measure of efficiency, since such times are strongly dependent upon the implementation of interval arithmetic, etc. However, we supplement table 1 with a list of total clock times on an unloaded IBM 3090 with the VM/CMS operating system; these appear in table 2. (Note that these times will change with system load, etc.)

The extra running times for our new algorithms in table 2 may be due partially to fluctuations in system load. However, they are probably also due to repeated application of the classical Newton's method in step 1 of algorithm 3.1. This problem can be remedied by storing the points X_* for later use, or by using a more finely tuned heuristic to determine singularity.

5. Summary, conclusions, and future work

We have considered interval Newton/generalized bisection algorithms as foolproof methods for solving the global optimization problem. We have developed theory and algorithmic details for techniques to make these algorithms more efficient when the Hessian matrix is either ill-conditioned or singular at the optimum. The algorithm has been written to find all critical points, but can be modified to efficiently find just the global optimum (and corresponding parameter set or sets).

The results of numerical experiments indicate that the techniques have value, and can possibly be applicable in a general code.

The algorithm can undoubtedly be further "tuned". In particular, the criterion in step 3(f) of algorithm 3.2 to determine when to execute algorithm 3.1 can possibly be made more appropriate. Also, Newton's method in algorithm 3.1 is applied redundantly, since it is unnecessary to compute the root again for any $\mathbf{X}_k^3$ which has been formed from some $\mathbf{X}_k$; considerable CPU time could be saved by storing the root X_* along with the other stack information associated with $\mathbf{X}_k^3$. Furthermore, alternate ways of determining whether the system is ill-conditioned (step 2 of algorithm 3.1), or different values of $\kappa_{\max}$, may lead to an algorithm which executes, on average, in less CPU time.

References

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

[2] E.R. Hansen, On solving systems of equations using interval arithmetic, Math. Comp. 22 (1968) 374–84.

[3] E. Hansen, Global optimization using interval analysis – the multidimensional case, Numer. Math. 34 (3) (1980) 247–70.

[4] E. Hansen, and S. Sengupta, Bounding solutions of systems of equations using interval analysis, BIT 21 (1981) 203–11.

[5] R.B. Kearfott, Abstract generalized bisection and a cost bound, Math. Comp. 49 (179) (1987) 187–202.

[6] R.B. Kearfott, Some tests of generalized bisection, ACM Trans. Math. Software 13 (3) (1987) 197–220.

[7] R.B. Kearfott, On handling singular systems with interval Newton methods, IMACS Ann. Comp. Appl. Math. 1.2 (1989) 653–655.

[8] R.B. Kearfott, Preconditioners for the interval Gauss–Seidel method, accepted for publication, SIAM J. Numer. Anal. 27 (3) (1990) 804–822.

[9] R.B. Kearfott, Interval arithmetic techniques in the computational solution of nonlinear systems of equations: introduction, examples, and comparisons, in: *Proc. 1988 AMS/SIAM Seminar on Applied Mathematics*, American Mathematical Society (1990) pp. 337–357.

[10] R.B. Kearfott, An interval step control for continuation methods, submitted, Math. Comp. (1989).

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

[12] R.E. Moore and S.T. Jones, Safe starting regions for iterative methods, SIAM J. Numer. Anal. 14 (6) (1977) 1051–065.

[13] A. Neumaier, Interval iteration for zeros of systems of equations, BIT 25 (1) (1985) 256–73.

[14] A. Neumaier, *Interval Methods for Systems of Equations* (Cambridge University Press, 1989; in press).

[15] K. Nickel, On the Newton method in interval analysis, Technical Summary Report no. 1136, Mathematics Research Center, the University of Wisconsin at Madison, Madison, WI.

[16] H. Ratschek and J. Rokne, *Computer Methods for the Range of Functions* (Ellis Horwood and John Wiley, Chichester, New York, 1984).

[17] H. Ratschek and J. Rokne, *New Computer Methods for Global Optimization*, (Ellis Horwood and John Wiley, Chichester, New York, 1988).

[18] J.M. Shearer and M.A. Wolfe, Some computable existence, uniqueness, and convergence tests for nonlinear systems, SIAM J. Numer. Anal. 22 (6) (1985) 1200–207.