1

A REVIEW OF TECHNIQUES IN THE VERIFIED SOLUTION OF CONSTRAINED GLOBAL OPTIMIZATION PROBLEMS

R. Baker Kearfott*

Department of Mathematics
University of Southwestern Louisiana
U.S.L. Box 4-1010, Lafayette, LA 70504-1010, USA
email: rbk@usl.edu

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

ABSTRACT

Elements and techniques of state-of-the-art automatically verified constrained global optimization algorithms are reviewed, including a description of ways of rigorously verifying feasibility for equality constraints and a careful consideration of the role of active inequality constraints. Previously developed algorithms and general work on the subject are also listed. Limitations of present knowledge are mentioned, and advice is given on which techniques to use in various contexts. Applications are discussed.

1 INTRODUCTION, BASIC IDEAS AND LITERATURE

We consider the constrained global optimization problem

$$\begin{array}{rlcll} \text{minimize} & \phi(X) & & & \\ \text{subject to} & c_i(X) & = & 0, \quad i = 1, \ldots, m & \\ & a_{i_j} \leq & x_{i_j} & \leq b_{i_j}, \quad j = 1, \ldots, q, & \end{array} \tag{1.1}$$

where $X = (x_1, \ldots, x_n)^T$. A general constrained optimization problem, including inequality constraints $g(X) \leq 0$ can be put into this form by introducing slack variables s, replacing by $s + g(X) = 0$, and appending the bound constraint $0 \leq s < \infty$; see §2.2.

We wish to find *all* minimizers in problem 1.1, and to *verify* bounds on the local minimum. Because of this, the methods used must go beyond (and build upon) traditional methods utilizing descent, line searches, trust regions, etc. in floating-point arithmetic, such as those in the books [7], [13] [46], or in the software package of [4]: these "approximate minimization" methods find only one approximate minimizer per run, may terminate near points other than minimizers without indications that they have done so, may converge to local minimizers that are not global minimizers, etc. The efficiency of such approximate methods, however, makes them practical. They can be used as in [22] and [24] or [3] to make verification algorithms efficient.

The conditions $a_{i_j} \leq x_{i_j} \leq b_{i_j}$, $j = 1, \ldots, q$ represent actual bound constraints intrinsic to the problem formulation. In rigorous branch and bound algorithms, an *overall search region* $\mathbf{X}_0$ is generally defined through similar bounds:

$$\mathbf{X}_0 = \{(x_{0,1}, \ldots, x_{0,n}) \mid a_i \leq x_{0,i} \leq b_i, \quad 1 \leq i \leq n\}; \tag{1.2}$$

only those bounds corresponding to the index set $\{i_j\}_{j=1}^{q}$ should be treated as actual bound constraints.

Deterministic location of the global minima and all global minimizers of the non-convex constrained optimization problem 1.1 is in general very difficult (and, indeed, NP-complete; cf. §2.2 and §2.3). However, interval branch and bound methods have exhibited a degree of success in many instances. Various authors, including Moore [43] and Skelboe [65], Hansen ([17], [18] and [19]), Ichida [21], Ratschek and Rokne [52], Jansson and Knüppel ([23] and [24]), Caprani, Godthaab and Madsen [3] have contributed to the knowledge of such algorithms. See [52] for a coherent explanation of this type of algorithm, as well as for the requisite introduction to interval arithmetic[1].

The basic ideas behind versions of such algorithms for unconstrained optimization (where $m = 0$) can be stated with minimal notation and without reference to details of interval arithmetic. The principles include a *check on the range* of ϕ and a *computational existence / uniqueness* test. We let $\mathbf{X} = (\mathbf{x}_1, \ldots \mathbf{x}_n)^T = ([\underline{x}_1, \overline{x}_1], \ldots, [\underline{x}_n, \overline{x}_n])^T$ be the interval vector ("box") representing the search region $\underline{x}_i \leq x_i \leq \overline{x}_i$, $1 \leq i \leq n$. Branch and bound algorithms maintain one or more of these lists: a list $\mathcal{L}$ of boxes $\mathbf{X}$ to be processed, a list $\mathcal{U}$ of boxes the algorithm has reduced to small diameter, and a list $\mathcal{C}$ of boxes that have been verified to contain critical points. The general pattern is as follows.

[1] Other texts, such as [1], [15], [19], [43] and [48] also give good introductions to the subject, and contain techniques relevant to global optimization.

Algorithm 1 (Abstract Branch-and-Bound Pattern)

1. *Initialize $\mathcal{L}$ by placing the initial search region $\mathbf{X}_0$ in it.*
2. DO WHILE $\mathcal{L} \neq \emptyset$.
   (a) *Remove the first*[2] *box $\mathbf{X}$ from $\mathcal{L}$;*
   (b) (Process $\mathbf{X}$) *Do one of the following:*
      - *reject $\mathbf{X}$;*
      - *reduce the size of $\mathbf{X}$;*
      - *determine that $\mathbf{X}$ contains a unique critical point, then find the critical point to high accuracy.*
      - *subdivide $\mathbf{X}$ to make it more likely to succeed at rejecting, reducing, or verifying uniqueness.*
   (c) *Insert one or more boxes derived from $\mathbf{X}$ onto $\mathcal{L}$, $\mathcal{U}$ or $\mathcal{C}$, depending on the size of the resulting box(es) from step 2b and whether the (possible) computational existence test in that step has determined a unique critical point.*

End Algorithm 1

Many details, such as stopping criteria and tolerances, are absent from Algorithm 1, which represents a general description. Such details differ in particular actual algorithms.

A combination of several techniques is used in state-of-the-art interval global optimization codes to do step 2b of Algorithm 1.

Algorithm 2 (Range Check and Critical Point Verification)

1. *Input a box $\mathbf{X}$ and the current best rigorous upper bound $\overline{\phi}$ on the global minimum.*
2. (Feasibility check; for constrained problems only)
   (a) (Exit if infeasibility is proven) DO *for $i = 1$ to m:*

[2]The boxes in $\mathcal{L}$ are in general inserted in a particular order, depending on the actual algorithm

i. Compute an enclosure $c_i(\mathbf{X})$ *for the range of* c_i *over* $\mathbf{X}$.

ii. IF $0 \notin c_i(\mathbf{X})$ THEN *discard* $\mathbf{X}$ *and* EXIT.

(b) Prove computationally, if possible, that there exists at least one feasible point in $\mathbf{X}$.

3. (Range check or " midpoint test")

(a) Compute a lower bound $\underline{\phi}(\mathbf{X})$ *on the range of* ϕ *over* $\mathbf{X}$.

(b) IF $\underline{\phi}(\mathbf{X}) > \overline{\phi}$ THEN *discard* $\mathbf{X}$ *and* EXIT.

4. (Update the upper bound on the minimum.) IF *the problem is unconstrained or feasibility was proven in step 2b,* THEN

(a) Use interval arithmetic to compute an upper bound $\overline{\phi}(\mathbf{X})$ *of the objective function* ϕ *over* $\mathbf{X}$.

(b) $\overline{\phi} \leftarrow \min\{\overline{\phi}, \overline{\phi}(\mathbf{X})\}$.

5. (" monotonicity test")

(a) Compute an enclosure $\nabla\phi(\mathbf{X})$ *of the range of* $\nabla\phi$ *over* $\mathbf{X}$. *(Note: If* $\mathbf{X}$ *is "thin", i.e. if some bound constraints are active over* $\mathbf{X}$*, then a* reduced gradient *can be used; see §2.3 and [32].)*

(b) IF $0 \notin \nabla\phi(\mathbf{X})$ THEN *discard* $\mathbf{X}$ *and* EXIT.

6. (" concavity test") *If the Hessian matrix*[3] $\nabla^2\phi$ *cannot be positive definite anywhere in* $\mathbf{X}$ THEN *discard* $\mathbf{X}$ *and* EXIT.

7. (Quadratic convergence and computational existence / uniqueness) *Use an interval Newton method*[4] *(with the Fritz–John conditions as in §3.5 in the constrained case) to possibly do one or more of the following:*

- *reduce the size of* $\mathbf{X}$*;*
- *discard* $\mathbf{X}$*;*
- *verify that a unique critical point exists in* $\mathbf{X}$.

8. (Bisection or geometric tessellation) *If step 7 did not result in a sufficient change in* $\mathbf{X}$*, then bisect* $\mathbf{X}$ *along a coordinate direction (or otherwise tessellate* $\mathbf{X}$*), returning all resulting boxes for subsequent processing.*

End Algorithm 2

[3] or reduced Hessian matrix, as in step 5

[4] possibly in a subspace, as in steps 5 and 6

Since techniques for constrained problems are somewhat more involved, step 2, checking for infeasibility and verifying feasibility, will be explained separately in §2.1.

Step 3 is called the " midpoint test" because the upper bound $\overline{\phi}$ is often obtained by evaluating ϕ at the midpoint vector of $\mathbf{X}$, properly taking account of rounding errors for rigor. Of course, step 5 is called the " monotonicity test " since ϕ is monotone over $\mathbf{X}$ in the i-th variable if the i-th component of $\nabla\phi$ does not vanish over $\mathbf{X}$.

Improved techniques for carrying out step 6, checking non-convexity, are desirable. Presently, sufficient conditions, such as checking the sign of the diagonal entries of an interval evaluation $\nabla^2\phi(\mathbf{X})$, can be used. One method of verifying convexity appears as Theorem 14.1 in [15] and Lemma 2.7.2 in [54]. Also, Neumaier [49] has shown that every element matrix of an interval matrix $\mathbf{A}$ is positive-definite, provided some point matrix $A \in \mathbf{A}$ is positive definite and $\mathbf{A}$ is regular according to Definition 3 of §3 below. This result can be sharpened as follows.

Theorem 1 (Shi, [64, Appendix B]) *If $A \in \mathbf{A}$ is symmetric, A has p negative eigenvalues and $\mathbf{A}$ is regular, then every symmetric point matrix in $\mathbf{A}$ has p negative eigenvalues.*

Theorem 1 will allow a sharper concavity test.

Interval global optimization algorithms, viewed abstractly, are similar to branch and bound algorithms that do not explicitly use interval arithmetic to bound ranges, such as that in [50, Ch. 6]. Interval global optimization algorithms differ among themselves in the ordering of the lists $\mathcal{L}$, $\mathcal{U}$ and $\mathcal{C}$ in step 2c of Algorithm 1, and in how (and which) steps of Algorithm 2 are carried out.

1.1 Early and simplified algorithms

Early algorithms worked only with the list $\mathcal{L}$, without lists $\mathcal{U}$ and $\mathcal{C}$. Also, although the processes in steps 3 through 7 of Algorithm 2 make actual implementations practical and efficient, they are not an essential part of the branch and bound structure. In the early but well-known Moore / Skelboe algorithm, only the list $\mathcal{L}$ appears, and the boxes $\mathbf{X} \in \mathcal{L}$ are ordered in order of increasing $\underline{\phi}(\mathbf{X})$. In step 8 of Algorithm 2, $\mathbf{X}$ is bisected along the largest coordinate

direction, and both progeny are placed in order in $\mathcal{L}$. Steps 3 through 7 of Algorithm 2 are absent. When the algorithm is terminated, the first box in the list is taken to approximate the global minimizer.

An algorithm attributed to Ichida [21] improves upon the Moore / Skelboe algorithm by including the midpoint test (step 3 of Algorithm 2) to avoid placing boxes generated during bisection onto $\mathcal{L}$ if they cannot contain optimizers. Additionally, the algorithm described in [21] contains a method for grouping together clusters of boxes corresponding to particular minimizers.

Hansen's algorithms, described in [17], [18] and [19] generally use second-order information (step 7 of Algorithm 2) and other sophisticated techniques. However an algorithm sometimes called " Hansen's algorithm" is a simplified version. In " Hansen's algorithm," $\mathcal{L}$ is ordered not in terms of the function, but in order of decreasing diameter (i.e. width of largest coordinate interval) of the $\mathbf{X}$. Furthermore, in the midpoint test, the entire list $\mathcal{L}$ is culled (and not just the boxes that have just been produced by bisection) whenever a new $\overline{\phi}$ is obtained. (We note that in Hansen's actual codes, as in the experiments in [66], the list is ordered such that the first box is the one with smallest lower bound on ϕ. Hansen and Walster claim this is much better than ordering in terms of decreasing diameter.) Various modifications of the list ordering, such as that in [54, §2.2.5.1], have appeared more recently.

None of these simplified methods employs interval-Newton acceleration.

Convergence properties of the Moore / Skelboe, Ichida and Hansen algorithms, as well as some numerical experiments with Hansen's algorithm are analyzed in [44]. A wide-ranging survey that concentrates on these methods (but also mentions some newer techniques) is the book [52].

1.2 Recent practical algorithms

More recent algorithms and practical implementations usually involve interval Newton methods for computational existence and uniqueness of critical points and for quadratic convergence properties. However, some successful newer algorithms are derivative-free, and concentrate on use of approximate optimizers, order in which the list is searched, properties of the inclusion function, or parallelization.

A thorough exposition of background, starting with the elements of interval arithmetic, and of numerous techniques for interval unconstrained optimization along with a substantial number of careful numerical experiments appears in Ratz [54]. Some of these ideas are implemented in the Pascal-XSC code described in [15].

Ratz, continuing development of his algorithms, has concentrated on better choice of coordinate in the bisection process of step 8 of Algorithm 2, and on *splitting strategies*, cf. [55]. Regarding bisection strategies, Ratz claims better success when choosing the coordinate to bisect according to the scaling

$$\nabla\phi(\mathbf{X})(\mathbf{X} - X),$$

rather than merely bisecting along the longest coordinate direction of $\mathbf{X}$; cf. [54], pp. 41–42 and [6]. Convergence of generalized bisection based global optimization algorithms with this coordinate selection strategy is also proven in [6]. This scheme is related to the *maximal smear* scheme introduced in [30]. Box splitting is a process, first discussed by Hansen in relation to the interval Gauss–Seidel method, by which extended interval arithmetic in the sense of Kahan is used to obtain two disjoint boxes. If applied wherever possible, too many boxes are produced, thus slowing the overall branch and bound algorithm. Coordinate choice in bisection, box-splitting strategies, and the ordering in the list $\mathcal{L}$ can be crucial in an overall global optimization algorithm.

Hansen's book [19] provides an informal description of many sophisticated techniques and heuristics for use in global optimization algorithms. Many examples are given, although thorough numerical experiments are absent. Good numerical experiments with an earlier optimization algorithm incorporating many of the same ideas appear in [66].

In [23] and [24], Jansson and Knüppel have presented a method without derivatives (no gradient test or interval Newton method), but with a sophisticated use of bisection and local optimization. In particular, a local optimization (to obtain an approximate optimizer) is performed at certain stages of the process, and the results are used to update $\overline{\phi}$. Though heuristic, the algorithm performs well on many reasonably complicated functions, including non-differentiable ones, such as maximizing the smallest singular value of a matrix. The report [24] also contains a collection of test examples, along with numerical results and three-dimensional graphs of those (numerous) test problems that are two-variable functions. Jansson [25, §2.5] proposes a variant in which derivatives

are used only in an interval Newton method to verify and sharpen bounds for approximate optima. This variant is carefully tested on forty test problems in [26].

In [3] Caprani, Godthaab, and Madsen also propose use of an approximate minimizer obtained through a local method with floating point arithmetic. In their algorithm, an approximate local minimizer is found, then a box $\mathbf{X}$ is constructed about this minimizer. An interval Newton method is then applied to $\mathbf{X}$ to determine existence or uniqueness of a critical point. If existence can be proven, $\mathbf{X}$ is expanded as much as possible, subject to success of the interval Newton method in verifying uniqueness, then $\mathbf{X}$ is removed from the region by cutting the complement of $\mathbf{X}$ into remaining boxes to be processed. The minimizer-inflation technique is related to "ϵ-inflation" as in [57, p. 58] or [41] used to provide error bounds on approximate solutions to linear and nonlinear systems. It is illustrated in [3] that the Caprani/Godthaab/Madsen method parallelizes well.

In [45], basic algorithms for non-differentiable and differentiable objective functions are reviewed, then a sophisticated coarse-grained algorithm for optimization on a distributed-memory multicomputer, implemented on a distributed system of workstations, is explained. In this algorithm, each processor shares a portion of the list $\mathcal{L}$. The load is dynamically balanced as the computations proceed. The algorithm was programmed in C++, based on a system for interval arithmetic developed by Leclerc; an encapsulated explanation appears in [40]. The numerical experiments feature a very difficult parameter-fitting problem.

In [10] and [11] Eriksson et al also study parallelization of an unconstrained global optimization algorithm, implemented on an Intel hypercube. Various load balancing strategies are compared on a set of six test problems, one of which was designed specifically to test different parallelization schemes.

In [32], experimental results are reported for a FORTRAN-77 code containing techniques for the monotonicity test and iteration / verification, as well as use of a local optimization process for computing $\overline{\phi}$ (" midpoint test "). The special preconditionerspreconditioning techniques of [29] and [31] for the interval Gauss–Seidel method (a type of interval Newton method) in the optimization context, as well as a technique for handling bound constraints through the tessellation are studied there[5]. A more flexible Fortran-90 code utilizing the

[5]The latter two techniques are more fully explored in [54].

Method / Authors	Midpoint Test	Monotonicity Test	Concavity Test	Interval Newton	Parallelization	Use of Local Minimizer	Ref.
Moore / Skelboe							[43] and [65]
Ichida	yes						[21]
"Hansen's algorithm"	yes, and to cull list						[52]
Hansen's actual	yes	yes	yes	yes			[19]
Kearfott '92	yes	yes		yes		yes	[32]
Ratz	yes	yes	yes	yes		yes	[54] and [15]
Jansson / Knüppel	yes				yes	yes	[24]
Caprani / Godthaab / Madsen	yes			yes	yes	yes	[3]
Hansen / Moore / Leclerc	yes	yes	yes	yes	subject of study		[45]
Eriksson	yes	yes		yes	subject of study		[11]
Wolfe	yes	yes		yes			[67]

Table 1 Summary attributes of various global optimization algorithms

system of [34] and including techniques for constrained problems is presently under development.

Theoretical and empirical consequences of the order of the interval extension used to obtain $\underline{\phi}(\mathbf{X})$ are studied in [8] and [37]. However, exhaustive studies on a practical algorithm do not appear there.

Some (but not all) of the attributes of the algorithms in this section and in §1.1 are summarized in Table 1. Here, the label "Kearfott '92" refers to the code of [32]. "Hansen's actual" is used to denote the most recent algorithms of Hansen, described in [19] and forming the basis of the work in [45] and [40]. Blank spaces in the table indicate that the feature is not present.

1.3 Notation

In the remainder of this paper, we will use the following notation. We will use boldface to denote intervals, lower case to denote scalar quantities, and upper case to denote vectors and matrices. We will use underscores to denote lower bounds of intervals and overscores to denote upper bounds of intervals. For components of vectors, we will use corresponding lower case letters. For example, we may have:

$$\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n]^T,$$

where $\mathbf{x}_i = [\underline{x}_i, \overline{x}_i]$. The notation $\check{X}$ will denote a representative point, usually in the interval vector $\mathbf{X}$. The *magnitude* of an interval is defined as $|\mathbf{x}| = \max\{|\underline{x}|, |\overline{x}|\}$.

The width of an interval $\mathbf{x}$ is denoted by $w(\mathbf{x}) = \overline{x} - \underline{x}$. The width of an interval vector $\mathbf{X}$, denoted $w(\mathbf{X})$, is defined component-wise. We use $w(\mathbf{X})$ in the context of $\|w(\mathbf{X})\| = \|w(\mathbf{X})\|_\infty$.

The symbol $\phi(\mathbf{X})$ denotes an interval extension of ϕ over $\mathbf{X}$.

Whenever $\|\cdot\|$ is used, it will mean $\|\cdot\|_\infty$.

We will use calligraphic letters such as $\mathcal{L}$, $\mathcal{U}$ and $\mathcal{C}$, as above, to denote stacks and lists of boxes.

Brackets $[\cdot]$ will be used to delimit both intervals and matrices and vectors. For example, we have the interval $[1, 2]$ and the interval vector $[[1, 2], [3, 4]]'$. Meanings should be clear from the context.

The notation $\text{int}(\mathbf{X})$ will be used to denote the topological interior of an interval vector $\mathbf{X}$.

Interval arithmetic will not be reviewed here, as numerous introductions, such as those in [1], [43], [48], [52] or even [28], exist.

We will also assume familiarity with general concepts of numerical methods for constrained and unconstrained optimization, such as are found in [13].

2 ON CONSTRAINED OPTIMIZATION PROBLEMS

The book [52] contains an explanation of fundamental interval means of handling inequality constraints, while [19] contains an in-depth treatment of many interval techniques for both inequality and equality constraints. However, extensive numerical results using these techniques have not been published. (See Table 2; blank spaces mean the feature is absent.)

Handling of simple bound constraints through the tessellation process has been explored in [32] and [54]. In general, the computational effort for this technique

Method / Authors	Bound Constraints	Inequality Constraints	Equality Constraints	Second Order	Use of approx. minimizer	Numerical Experiments
"Hansen's"						unconstrained
Hansen's book	(as equality constraints)	yes	yes	yes		unconstrained only
Opt. '92	yes				yes	yes
Ratz	yes					yes
Jansson / Knüppel	yes			variant in [25, §2.5]	yes	unconstrained only
Caprani / Godthaab / Madsen					yes	unconstrained
Hansen / Moore / Leclerc						unconstrained only
Eriksson						unconstrained
Wolfe		yes	yes	yes		yes

Table 2 Summary of handling of constraints in various global optimization algorithms.

increases exponentially with the number of bound constraints, but this may not be so for many specific problems. However, bound-constrained problems are intrinsically hard [51]. Although bound constraints can be handled as inequality constraints as in [19], it is unclear without published experimentation how such algorithms behave: it is possible that large numbers of small boxes, clustered on the boundaries of the constraints, are produced through the n-dimensional tessellation.

Alternately, inequality constraints can be handled as bound constraints by introduction of slack variables.

We elaborate on these concepts in the remainder of this section.

2.1 Checking Feasibility / Infeasibility

The following computations may be done with the constraints:

- using the constraints to delete portions of a region $\mathbf{X}$ that are infeasible;
- proving feasibility or infeasibility relative to inequality constraints;
- proving feasibility or infeasibility relative to equality constraints.

These possibilities are discussed in the remainder of this section.

In [19, §11.6], Hansen proposes heuristics for using inequality constraints to delete portions of a box $\mathbf{X}$ that cannot be feasible. Alternately, if the inequality constraints are converted into equality constraints first, the optimal preconditioner techniques of [29] and [31] in conjunction with the interval Gauss–Seidel method or interval Gaussian elimination may be used directly on the underdetermined m by n system of constraints. The latter provides a certain theoretical optimality not present in the heuristics of [19], with a smaller system than with the entire Fritz–John system; this appears in [36]. However, experiments in [36] indicate that the scheme reported in this section is usually better.

An algorithm for constructing large boxes within which inequality constraints of the form $g(x) < 0$ are rigorously verified appears in [39], along with numerical experimentation. We have not included this algorithm in our tables, however, since it is not a general global search algorithm, but a method of dealing with constraints.

As indicated in step 2a of Algorithm 2, an elementary check for infeasibility of an entire box $\mathbf{X}$ with respect to the equality constraints c_i is to verify that $0 \notin c_i(\mathbf{X})$ for some i. There is a corresponding check if an inequality constraint $d(x) \leq 0$ is used: the region is infeasible if simply $d(\mathbf{X}) > 0$. Similarly, if only inequality constraints of the form $d_i \leq 0$ are present, feasibility is proven if $d_i(\mathbf{X}) \leq 0$ for each i. In this case, Hansen calls the region *certainly feasible*.

On the other hand, proving feasibility for problems with equality constraints $c_i(x) = 0$ (like problem 1.1), although necessary to get useful rigorous upper bounds $\overline{\phi}$ on the global minimum, is more difficult. In general, unless the number of constraints m is less than the number of variables n, the problem is infeasible unless the constraints are linearly dependent. Walster and Hansen have mentioned work (in private communication) related to handling dependent constraints, but we are unaware of published material in this area. See [36] for some thoughts on dependent constraints; we will assume independent constraints and $m \leq n$ here.

In [67], Wolfe proposes an algorithm, based on a penalty function, for handling equality constraints. However, that algorithm considers feasibility to be rigorously proven only provided $c_i(x) \in [\epsilon, \epsilon]$ for each i and some fixed ϵ. In contrast, in a method proposed in Hansen [19, §12.3 ff] and in [36], $c_i(x) = 0$ is rigorously proven. Both these methods are based on applying an interval Newton method to the system $c_i(x) = 0$, $i = 1, \ldots, m$ with $n - m$ of the variables held fixed. However, the methods differ in application of the interval Newton

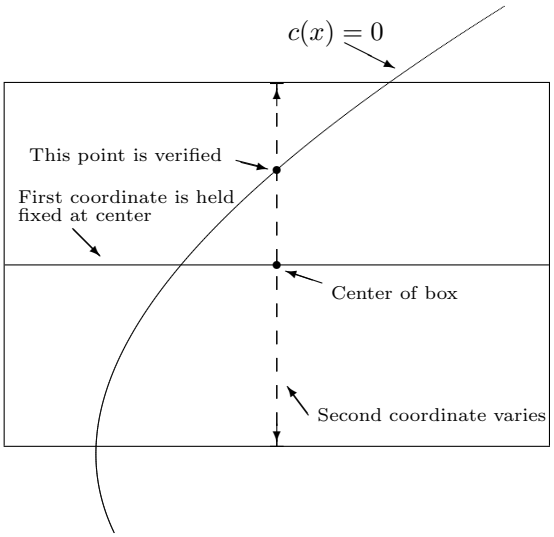


Figure 1 Verifying a feasible point with Algorithm 3

method and in the choice variables to be held fixed. For details and numerical comparisons, see [36].

The following general algorithm encompasses Hansen's and our proposed feasibility verification methods of [19] and [36]; the techniques differ in the details of step 3.

Algorithm 3 (Prove Feasibility for Equality-Constrained Problems)

1. *Input an approximation $\check{X}$ to a feasible point, obtained through a conventional algorithm such as that of [4].*
2. *Let $C(X) = (c_1(X), \ldots c_n(X))^T$ and $\nabla \mathbf{C}(X)$ represent the equality constraints and Jacobi matrix of the equality constraints, respectively ($C : \mathbb{R}^n \to \mathbb{R}^m$).*
3. *Choose coordinates $\{p_k\}_{k=1}^m$ to be varied in the interval Newton method in such a way that the resulting system is likely to be nonsingular.*
4. *Evaluate $C(\check{X})$ and $\nabla \mathbf{C}(\hat{\mathbf{X}})$, where $\hat{\mathbf{X}}$ has coordinates $\hat{\mathbf{x}}_i = \check{x}_i$ if $i \neq p_k$ for any k and $\hat{\mathbf{x}}_i = [\check{x}_i - \epsilon, \check{x}_i + \epsilon]$ for $i = p_k$ for some ϵ. That is, construct a box*

in an m-dimensional subspace that is, in a sense, most nearly perpendicular to the null space of $\nabla C(\check{X})$.

5. *Let* Y *be an approximate inverse of the* $m \times m$ *matrix consisting of columns* p_1 *through* p_m *of the midpoint matrix of* $\nabla \mathbf{C}(\hat{\mathbf{X}})$.
6. *Apply an interval Newton method to* $Y \nabla \mathbf{C}(\hat{\mathbf{X}})(X - \check{X}) = -C(\check{X})$, *obtaining an image* $\hat{\mathbf{X}}_{new}$.
7. *If the result* $\hat{\mathbf{X}}_{new} \subseteq \text{int}(\hat{\mathbf{X}})$, *then the point* $\check{X}$ *is feasible. Find* $\phi(\check{X})$, *and take the upper bound as a rigorous upper bound for a minimum.*

End Algorithm 3

Here, the interval Newton method of step 6 may be taken as an operator $\mathbf{N}(\hat{\mathbf{X}}, F)$ operating on a function $F : \mathbb{R}^p \to \mathbb{R}^p$ defined on a box $\hat{\mathbf{X}}$, such that

1. the image $\mathbf{N}(\mathbf{X}, F)$ is a box in $\mathbb{R}^p$;
2. if $\mathbf{N}(\mathbf{X}, F)$ is contained in the interior of $\mathbf{X}$, then there is a unique solution to $F(X) = 0$ within $\mathbf{X}$;
3. $\mathbf{N}(\mathbf{X}, F)$ is straightforward to compute with interval arithmetic in an appropriate software environment.

See §3 for advice on interval Newton methods and for pointers to the abundant literature.

Full experiments on this algorithm, as well as comparisons, are reported in [36]. The general conclusions are that Algorithm 3 is effective, and that step 3 is effective, *provided a good approximation to a feasible point is centered in a box whose center is at* $\check{X}$. For a detailed explanation and additional illustrations, see [36].

2.2 On Equality, Inequality and Bound Constraints

The book [52] poses the optimization problem analogous to problem 1.1 with both equality constraints $c_i(x) = 0$ and inequality constraints $d_i(x) \leq 0$, while

[19] contains separate chapters on inequality constrained problems and equality constrained problems. At first glance, inequality constrained problems seem easier than equality-constrained ones. This is because feasibility can sometimes be proven without use of an interval Newton method: We merely bound the range of each d_j using interval arithmetic, then check that the upper bounds so obtained are all negative, i.e. by checking $\mathbf{d}_j(\mathbf{X}) \leq 0$ for each j. Also, besides the sophisticated technique in [19, §11.6], similar verification that $\mathbf{d}_j(\mathbf{X}) > 0$ for each j proves infeasibility over all of $\mathbf{X}$, allowing $\mathbf{X}$ to be eliminated from the global search region. Such techniques *should* probably be used in practical algorithms as additional tools to verify feasibility and to eliminate subregions. However, in such analyses, it is ignored that the inequality constraints can be *active*, that is, that they are in effect equality constraints.

Various theoretical results have been published in recent years showing that global optimization problems containing inequality constraints are NP-complete in the number of constraints. For example, it is shown in [51] that quadratic programming problems with one negative eigenvalue are NP-complete.

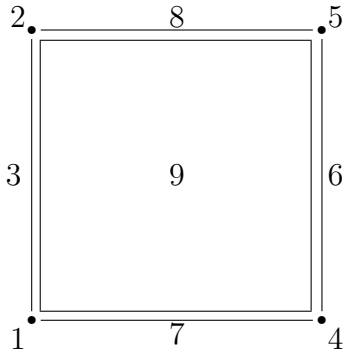
The possibility that computational effort can increase exponentially in the number of constraints becomes apparent if we examine the algorithm for bound-constrained problems in [32], reviewed in §2.3 below. However, as explained in §2.3, it is possible for heuristics to reduce the running time for specific problems below that predicted by the exponential worst-case bounds, without compromising rigor.

2.3 Handling Bound Constraints

In [32] and here, the bounds $a_{i_j} \leq x_{i_j} \leq b_{i_j}$ in problem 1.1 are viewed as bound constraints[6]. The bound constraints are then handled by separating the region into all possible subregions of lower dimensions, as is illustrated in Figure 2 for $n = 2$ These subregions are placed on the list $\mathcal{L}$ and processed as usual, except that reduced gradients and reduced Hessian matrices[7] are used in the interval Newton method on lower-dimensional regions. If all boxes are stored in $\mathcal{L}$, it is not difficult to see that the total number of boxes of all dimensions so obtained is 3^p, where p is the number of bound constraints. However, if a good upper bound $\overline{\phi}$ on the global minimum (as can sometimes

[6] in contrast to, say, [66], where these bounds are viewed as merely defining a search region in an unconstrained problem

[7] i.e. with rows and columns corresponding to variables held fixed deleted



There are 1 box in $\mathbb{R}^2$, 4 boxes in $\mathbb{R}^1$, and 4 boxes in $\mathbb{R}^0$.

Figure 2 "Peeling" a box to produce lower-dimensional boundary elements

be obtained with conventional algorithms such as that of [4]) is available, then many of the boxes can be rejected during the " peeling" process.

The structure of the algorithm for producing the list of lower-dimensional boxes can be described simply in recursive form. We have:

Algorithm 4 ("Peeling" the Boundary)

1. *Input the present coordinate index i, the box $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_n)$, the list $\mathcal{I}$ of those coordinate indices for $\mathbf{X}$ that have already been considered, and the list $\mathcal{L}$ of boxes that have been already been generated for further processing.*
2. IF $i \notin \mathcal{I}$ THEN
   - *(a)* (Process the lower boundary box.)
     - *i. Set all coordinates of $\tilde{\mathbf{X}}$ but the i-th to corresponding coordinates of $\mathbf{X}$. Set the i-th coordinate of $\tilde{\mathbf{X}}$ to $\underline{x}_i$.*
     - *ii. Set the index list $\mathcal{I}_{\text{new}}$ to $\mathcal{I}$ with i appended.*
     - *iii.* IF $i = n$
       THEN *store $\tilde{\mathbf{X}}$ in $\mathcal{L}$.*
       ELSE *execute this algorithm with $i+1$, $\tilde{\mathbf{X}}$, and $\mathcal{I}_{\text{new}}$ replacing i, $\mathbf{X}$, and $\mathcal{I}$, respectively.*
       END IF
   - *(b)* (Process the upper boundary box.)

i. Set all coordinates of $\tilde{\mathbf{X}}$ *but the* i*-th to corresponding coordinates of* $\mathbf{X}$*. Set the* i*-th coordinate of* $\tilde{\mathbf{X}}$ *to* $\overline{x}_i$*.*

ii. Set the index list $\mathcal{I}_{\text{new}}$ *to* $\mathcal{I}$ *with* i *appended.*

iii. IF $i = n$
THEN *store* $\tilde{\mathbf{X}}$ *in* $\mathcal{L}$*.*
ELSE *execute this algorithm with* $i+1$*,* $\tilde{\mathbf{X}}$*, and* $\mathcal{I}_{\text{new}}$ *replacing* i*,* $\mathbf{X}$*, and* $\mathcal{I}$*, respectively.*
END IF

(c) (Process the interior box.)

i. Set $\tilde{\mathbf{X}}$ *to* $\mathbf{X}$*.*

ii. Set the index list $\mathcal{I}_{\text{new}}$ *to* $\mathcal{I}$ *with* i *appended.*

iii. IF $i = n$
THEN *store* $\tilde{\mathbf{X}}$ *in* $\mathcal{L}$*.*
ELSE *execute this algorithm with* $i+1$*,* $\tilde{\mathbf{X}}$*, and* $\mathcal{I}_{\text{new}}$ *replacing* i*,* $\mathbf{X}$*, and* $\mathcal{I}$*, respectively.*
END IF

End Algorithm 4

The numbering of the nine boxes of dimensions 2, 1, and 0 in Figure 2 represents the order they would appear in $\mathcal{L}$ if each box generated with $i - n$ in steps 2(a)iii, 2(b)iii, and 2(c)iii of Algorithm 4 were stored. The processing order in Algorithm 4 can be viewed as traversing a ternary tree, as in Figure 3. The levels of this tree correspond to the coordinates i, with the root at $i = 1$ at the top and the leaves at $i = n$ at the bottom. As drawn in Figure 3, the order the leaves eventually appear in $\mathcal{L}$ is from left to right.

Of course, actual implementations of Algorithm 4 have additional steps to

- eliminate the boxes $\mathbf{X}$ and $\tilde{\mathbf{X}}$ before further processing or storage in $\mathcal{L}$ by checking $\phi(\mathbf{X})$ or $\phi(\tilde{\mathbf{X}})$ and the reduced gradient of ϕ on $\mathbf{X}$ or $\tilde{\mathbf{X}}$;
- skip coordinates i for which the bounds $a_i \le x_i \le b_i$ represent the extent of the search region, and not actual bound constraints for the problem.

These steps have been left out of the presentation of Algorithm 4 for clarity in the geometric process. However, they could be indispensable in reducing the number of boxes in $\mathcal{L}$ to a practical number. Observe that such steps can prune the tree in Figure 3 at a high level.

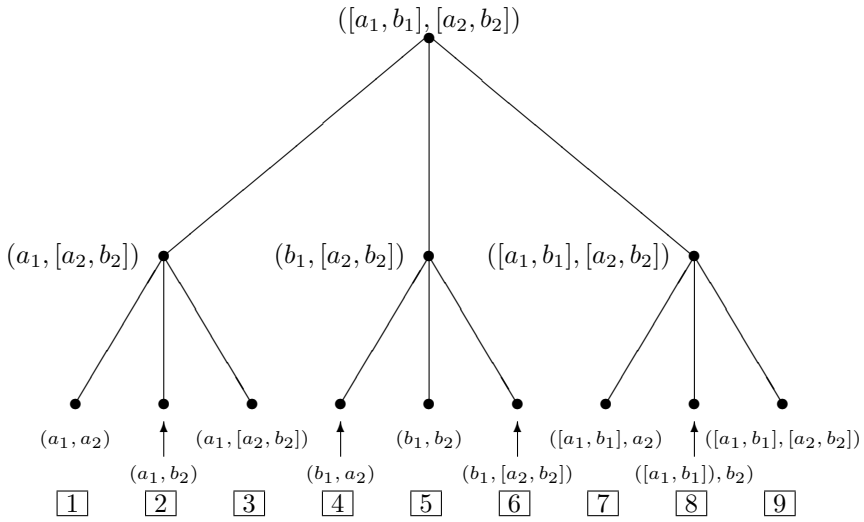


Figure 3 "Peeling" the box into lower-dimensional boundary elements

3 ON USE OF INTERVAL NEWTON METHODS

An interval Newton method is used in step 6 of Algorithm 2, and also must be used to verify feasibility in step 6 of Algorithm 3. Thus, interval Newton methods are used in the analysis of nonlinear algebraic systems arising in global optimization

- to reduce the size of regions $\mathbf{X}$, with quadratic convergence of the widths to zero;
- to computationally (but rigorously) prove existence of solutions within $\mathbf{X}$;
- to prove that there is a unique solution within $\mathbf{X}$;
- to prove that there can be no solutions in $\mathbf{X}$.

The basics of interval Newton methods are described in [1, Ch. 19], [43, Ch. 5 and Ch. 6], [52, §2.9], [19, Ch. 4 and Ch. 8], and numerous papers, with extensive theoretical development reviewed in [48, Ch. 4 and Ch. 5]. Here

we will concentrate on incorporating interval Newton methods into global optimization algorithms.

As mentioned in §2.1, an interval Newton method generally is an operator $\mathbf{N}(\mathbf{X}, F)$ associated with a function $F : \mathbb{R}^p \to \mathbb{R}^p$ defined on a box $\mathbf{X}$. Although actual derivations differ, we can intuitively think of interval Newton methods in terms of bounding the solution set to an interval linear system

$$\mathbf{A}(X - \check{X}) = -F(\check{X}). \tag{1.3}$$

The point $\check{X}$ is the midpoint of the box $\mathbf{X}$, an approximation to a solution to $F(X) = 0$, or some point that is otherwise appropriately chosen (usually, but not always, such that $\check{X} \in \mathbf{X}$). The interval matrix $\mathbf{A}$ can be chosen to be a *Lipschitz matrix* or a *slope matrix* for F; cf. [48] and definitions 1 and 2 in §3.3 below. In particular, an interval evaluation of the Jacobi matrix of F will yield an $\mathbf{A}$ that is a Lipschitz matrix. This is appropriate for many purposes.

The interval Newton method is not applied directly to equation (1.3), but to the preconditioned equation

$$Y\mathbf{A}(X - \check{X}) = -YF(\check{X}), \tag{1.4}$$

where Y is a p by p floating point matrix making the solution set to equation (1.4) easier to bound than that of equation 1.3. However, the solution set to equation (1.3) is in general a subset of the solution set to equation (1.4); cf. [48, §4.1].

Various ways of bounding the solution set to equation (1.3) or equation (1.4) appear in the literature. The interval hull, the smallest box or interval vector containing the solution set, can in principle be computed precisely. However, it may be impractical to compute the hull in practice, because the computation is NP complete in the order p of the nonlinear system; cf. [56].

There are three classes of methods in common use to compute interval bounds on the solution set to equation (1.3) or equation (1.4). These are the Krawczyk method, the interval Gauss–Seidel method, and interval Gaussian elimination.

Let $\tilde{\mathbf{X}}$ represent the bounds obtained on the solution set to equation (1.4). Then, omitting some technical conditions[8] on the interval Newton methods, we have the following.

1. Let $\mathbf{A}$ be a slope matrix for $\mathbf{X}$ based at $\check{X}$ or a Lipschitz matrix over $\mathbf{X}$. If $\tilde{\mathbf{X}} \subset \text{int}(\mathbf{X})$ then there exists a solution of $F = 0$ in $\mathbf{X}$.
2. Let $\mathbf{A}$ be a slope matrix for $\mathbf{X}$ based at $\check{X}$ or a Lipschitz matrix over $\mathbf{X}$. If $\tilde{\mathbf{X}} \cap \mathbf{X} = \emptyset$ then there are no solutions to $F(X) = 0$ in $\mathbf{X}$.
3. Let $\mathbf{A}$ be a Lipschitz matrix over $\mathbf{X}$. If $\tilde{\mathbf{X}} \subset \text{int}(\mathbf{X})$ then there exists a unique solution of $F = 0$ in $\mathbf{X}$.
4. If $\mathbf{A}$ is a slope matrix for $\mathbf{X}$ based at $\check{X}$ or a Lipschitz matrix over $\mathbf{X}$, then any root of F in $\mathbf{X}$ is also in $\tilde{\mathbf{X}}$. Furthermore, under certain conditions, replacing $\mathbf{X}$ by $\tilde{\mathbf{X}}$ leads to quadratic convergence of the widths of the components of $\mathbf{X}$ to zero. This allows subregions to be rigorously searched without excessive tessellation.

In addition, a compound algorithm can be used to show uniqueness when $\mathbf{A}$ is merely a slope matrix, and not a Lipschitz matrix. See [58] and the implementation description and experiments in [35].

Thus, performance of an interval Newton method can have differing goals (existence, uniqueness, nonexistence, or reduction in the size of $\mathbf{X}$). Also, interval Newton methods can be applied to different types of systems (general nonlinear system, Lagrange multiplier or Fritz–John system, etc.) Based on these and other considerations, different interval Newton methods are constructed. In summary, the items that can be varied are

- choice of method (direct hull computation, interval Gauss–Seidel, Krawczyk method, or interval Gaussian elimination);
- choice of preconditioner (if any);
- choice of the matrix $\mathbf{A}$ (slope matrix or Lipschitz matrix, and how it is computed);
- choice of the base point $\check{X}$.

We consider these choices in the context of global optimization algorithms in the remainder of this section.

[8] cf. [48] and more recent papers

3.1 Choice of Method

Algorithms for computation of the smallest interval vector containing the solution set to equation (1.3) are reviewed in [48, Ch. 6], while Rohn (who developed much of the theory of hull computation) reviews NP-hardness results in [56]. To our knowledge, computation of the exact interval hull is not used in any implementations of global optimization algorithms.

The Krawczyk method [38],

$$\mathbf{K}(\mathbf{X}, \check{X}) = \check{X} - YF(\check{X}) + [I - Y\mathbf{A}]\,(\mathbf{X} - \check{X}). \tag{1.5}$$

appears most frequently in the literature, especially in early work. A good analysis of the verification and convergence properties of the Krawczyk method appears in [42], and it appears as *the* interval Newton method in [43, Ch. 5]. It is known that the Krawczyk method is not as sharp as the interval Gauss–Seidel method $\mathbf{GS}(\mathbf{X}, \check{X})$, i.e. that $\mathbf{GS}(\mathbf{X}, \check{X}) \subseteq \mathbf{K}(\mathbf{X}, \check{X})$, for a given preconditioner Y. However, convergence analysis is easier with the Krawczyk method. Furthermore, Rump, as reviewed in [58], has developed a method of both *inner* and *outer* estimations to the solution set of equation (1.3) for the Krawczyk method, and thus rigorous computational bounds on the overestimation of the hull by $\mathbf{K}(\mathbf{X}, \check{X})$. Such estimates, and hence the Krawczyk method, are useful when the widths of the components of $\mathbf{X}$ are small.

The interval Gauss–Seidel method shares properties with the classical Gauss–Seidel method. Its iteration scheme is best described with an algorithm.

Algorithm 5 (Preconditioned interval Gauss–Seidel method)

DO *for* $i = 1$ *to* p.

1. (Update a coordinate) *Compute the* i*-th row* Y_i *of the preconditioner.*
2. (Do the step) *Let* $\mathbf{A}_j$ *denote the* j*-th row of* $\mathbf{A}$*. Compute*

$$\tilde{\mathbf{x}}_i = \check{x}_i - \left[Y_i F(\check{X}) \;+\; \sum_{j=1}^{i-1} Y_i \mathbf{A}_j (\tilde{\mathbf{x}}_j - \check{x}_j) \;+\; \sum_{j=i+1}^{p} Y_i \mathbf{A}_j (\mathbf{x}_j - \check{x}_j) \right] \Big/ (Y_i \mathbf{A}_i).$$

3. IF $\tilde{\mathbf{x}}_i \cap \mathbf{x}_i = \emptyset$,
THEN
EXIT, *signaling no solution.*
ELSE
Replace $\tilde{\mathbf{x}}_i$ *by* $\tilde{\mathbf{x}}_i \cap \mathbf{x}_i$.
END IF

END DO

End Algorithm 5

If Algorithm 5 does not terminate in step 3, then we label its output

$$\mathbf{GS}(\mathbf{X}, \check{X}) = \tilde{\mathbf{X}} = [\tilde{\mathbf{x}}_1, \ldots, \tilde{\mathbf{x}}_p]^T. \tag{1.6}$$

The analogous interval Jacobi method (each $\tilde{\mathbf{x}}_j$ is replaced by $\mathbf{x}_j$) and asynchronous parallel variants (each $\tilde{\mathbf{x}}_j$ is computed separately and made available for computations for the other j in an unpredictable way) are also worthy of consideration.

For a fixed preconditioner matrix Y, the image under the Gauss–Seidel method is at least as narrow (and hence, at least as good) as the image under the Krawczyk method. Furthermore, a set of optimal preconditioners, in the sense that the widths of the coordinates of the image are as small as possible, has been developed; cf. §3.2.

The third method, interval Gaussian elimination, sometimes gives better and sometimes worse results than the Krawczyk method or the interval Gauss–Seidel method; cf. [48, p. 158ff]. Although similar to the floating point Gaussian elimination algorithm, the *interval Gauss algorithm* is worth stating. In particular, although operations are done on intervals with non-zero widths during the elimination phase, it is rigorous to replace the elements being eliminated by exactly zero. Also, although the floating point Gauss algorithm has many variants, all of which are mathematically equivalent in exact arithmetic, these variants are not all the same when interval arithmetic is used. For this reason, we present the interval Gauss algorithm useful in our present context, as in [1, Ch. 15].

Algorithm 6 (Interval Gaussian Elimination)

1. (Input the matrix and right-hand side)
Input $Y\mathbf{A} = \mathbf{G} = [\mathbf{g}_{i,j}]$ *and*[9] $-YF(\check{X}) = \mathbf{S} = [\mathbf{s}_1, \ldots, \mathbf{s}_n]^T$.

2. (Factorization phase) DO *for* $i = 1$ *to* p*:*

(a) IF $0 \in \mathbf{g}_{i,i}$ THEN EXIT *with failure.*

(b) DO *for* $j = i + 1$ *to* p *and for* $k = i + 1$ *to* p*:*

$$\mathbf{g}_{j,k} \leftarrow \mathbf{g}_{j,k} - (\mathbf{g}_{j,i}/\mathbf{g}_{i,i})\,\mathbf{g}_{i,k}.$$

END DO

(c) $\mathbf{s}_i \leftarrow \mathbf{s}_i - (\mathbf{g}_{j,i}/\mathbf{g}_{i,i})\,b_i$

END DO

3. (Solution phase)

(a) $\mathbf{s}_{p,p} \leftarrow \mathbf{s}_p/\mathbf{g}_{p,p}$.

(b) DO *for* $i = p - 1$ *to* 1 *by* -1*:*

$$\mathbf{s}_i \leftarrow \left(\mathbf{s}_i - \textstyle\sum_{j=i+1}^{p} \mathbf{g}_{i,j}\mathbf{s}_j\right) \Big/ \mathbf{g}_{i,i}$$

END DO

4. $\tilde{\mathbf{x}}_i \leftarrow \mathbf{s}_i + \check{x}_i$ *for* $i = 1$ *to* p.

End Algorithm 6

The main advantage of the interval Gauss algorithm is that *starting bounds* $\mathbf{X}_0$ *are not required, except to compute the matrix* $\mathbf{A}$. This is a crucial property used in the method proposed in [20] and [19, §10.6] for bounding Lagrange multipliers, and hence for applying interval Newton methods to constrained problems; see §3.5 below.

Pivoting is generally not done in the interval Gauss algorithm, but the preconditioner matrix Y is designed so that $\mathbf{G}$ is like an identity matrix; see §3.2. However, Algorithm 6 can fail when the widths of the elements of $\mathbf{A}$ are large, when the preconditioner matrix Y is ill-conditioned, or when some combination of these conditions occurs. Even when the Gauss algorithm fails, it may still be possible to use the partial factorization to reduce some coordinate widths, provided initial estimates $\mathbf{X}_0$ for the coordinate intervals are available.

[9] Although $-YF(\check{X})$ is a point, $\mathbf{S}$ is transformed into an interval vector in this algorithm.

3.2 Choice of Preconditioner

The most commonly seen preconditioner Y in the Krawczyk, interval Gauss–Seidel, and interval Gauss algorithms is the approximate inverse of the matrix formed from the midpoints of the elements of $\mathbf{A}$, termed the *inverse midpoint preconditioner*. With this preconditioner, $\mathbf{G} = Y\mathbf{A}$ is[10] symmetric about the identity matrix. In certain contexts [58], this symmetry can be used to replace interval arithmetic by appropriately rounded floating point arithmetic. The preconditioner also has certain optimality properties that have been discussed in the literature in relation to all three methods. It is relatively simple to compute, its effect is relatively easy to analyze, and it is effective when the widths of $\mathbf{A}$ are small.

In [29], preconditioner rows for the interval Gauss–Seidel method based upon certain *width optimality* conditions were proposed. Since their computation requires solution of linear programming problems, these preconditioners are termed *optimal LP preconditioners*. Alternate optimality conditions and reformulation to a smaller LP problem appeared in [31]. The goal in [29] and [31] was to find Y so that the widths of the components $\tilde{\mathbf{X}}$ were minimized, and thus enable or speed convergence of interval iteration to a point.

As seen in [29], LP preconditioners can be effective provided the structure of the corresponding LP problems is utilized in their solution. Otherwise, the cost of solving these problems could overwhelm the advantages of an optimal preconditioner. It is still uncertain how useful these preconditioners can be in optimization. They are most advantageous when the relative widths of elements in different rows of $\mathbf{A}$ differ widely.

Optimal LP preconditioners as originally formulated may not be appropriate for verification. This is because minimization of $w(\tilde{\mathbf{x}}_i)$ may not be consistent with $\tilde{\mathbf{x}}_i \subset \text{int}(\mathbf{x}_i)$. However, it is possible to formulate the LP problem in terms of maximization of the possibility that $\tilde{\mathbf{x}}_i \subset \text{int}(\mathbf{x}_i)$. It is also possible to formulate optimal preconditioners for interval Gaussian elimination, and even for general systems[11] $\mathbf{A}X = B$. The inverse midpoint preconditioner is applicable to such systems while some such formulations lead to larger and hence more expensive-to-solve linear programming problems. Such formulations have not yet been fully investigated.

[10] approximately, if Y is an approximate inverse

[11] The interval Newton equation (1.3) is special because both the right-hand-side and the unknown $X - \tilde{X}$ approximate the zero vector.

3.3 Choice of Derivative Matrix

The matrices $\mathbf{A}$ for equation (1.3) generally satisfy one of the following two properties.

Definition 1 *(* [48, p. 174], etc.) *The matrix* $\mathbf{A}$ *is said to be a* Lipschitz matrix *for* F *over* $\mathbf{X}$ *provided, for every* $X \in \mathbf{X}$ *and* $Y \in \mathbf{X}$, $F(X) - F(Y) = A(X - Y)$ *for some* $A \in \mathbf{A}$.

Generally, uniqueness can be proven with an interval Newton method provided Y is nonsingular and $\mathbf{A}$ is a Lipschitz matrix; cf. [48].

An alternate, weaker property commonly discussed in the literature is the following.

Definition 2 *The matrix* $\mathbf{A}$ *is said to be an* interval slope matrix *for* F *over* $\mathbf{X}$ *and centered on the interval vector* $\check{\mathbf{X}}$, *provided, for every* $X \in \mathbf{X}$ *and* $\check{X} \in \check{\mathbf{X}}$, $F(X) - F(\check{X}) = A(X - \check{X})$ *for some* $A \in \mathbf{A}$.

Note that every Lipschitz matrix is a slope matrix, but not visa versa. As seen below, slope matrices can be computed whose entries are narrower than computed Lipschitz matrices.

The following concept is also relevant.

Definition 3 *An interval matrix* $\mathbf{A}$ *is said to be* regular *provided every* $A \in \mathbf{A}$ *is nonsingular.*

For example, regularity of $\mathbf{A}$ is implied by $\tilde{\mathbf{X}} \subset \mathbf{X}$ in equation (1.3) or equation (1.4). See [58] and [63] for methods in which values or ranges of F, but only the elements of $\mathbf{A}$, appear in the computational formula.

We will illustrate various choices for $\mathbf{A}$ by considering the second-order information for the objective function $\phi(x_1, x_2) = x_1^4/12 + x_1(x_2^3/3) + x_1^2/2 + x_2^2/2$, whose Hessian matrix is

$$\nabla^2 \phi(x_1, x_2) = \begin{bmatrix} x_1^2 + 1 & x_2^2 \\ x_2^2 & 2x_1x_2 + 1 \end{bmatrix}. \tag{1.7}$$

In these examples, F denotes $\nabla\phi$, so the derivative matrix $\mathbf{A}$ of F corresponds to the Hessian matrix $\nabla^2\phi$. We will examine matrices $\mathbf{A}$ for ϕ valid over the interval vector $\mathbf{X} = ([-.5, .5], [-.5, .5])^T$.

Using interval arithmetic to evaluate equation (1.7), we obtain:

$$[\text{Interval Hessian}] = \begin{bmatrix} [1, 1.25] & [0, .25] \\ [0, .25] & [.5, 1.5] \end{bmatrix}. \tag{1.8}$$

The interval Hessian matrix is a Lipschitz matrix for F over $\mathbf{X}$. It can thus be used flexibly in interval Newton algorithms. For example, when it is used to bound the solution set to equation (1.4) using the Krawczyk, interval Gauss–Seidel, or interval Gauss algorithms, uniqueness is implied by $\tilde{\mathbf{X}} \subset \text{int}(\mathbf{X})$, regardless of the choice of base point $\check{X}$, as long as $\check{X} \in \mathbf{X}$. The drawback is that the widths of the entries of an interval Hessian matrix may be so large that the resulting image box $\tilde{\mathbf{X}}$ is not useful.

In [16] and later in [19, §6.3-6.4], Hansen observes that, when evaluating the Hessian matrix for interval Newton methods, not all entries need be intervals. In particular, the j-th component of $\mathbf{X}$ in evaluation of $\partial^2\phi(X)/\partial x_i \partial x_k$ in the i-th row[12] of the matrix may be replaced by the j-th component of $\check{X}$ for $j > k$. Theory of uniqueness verification using this scheme has not been published. The matrix $\mathbf{A}$ so obtained is no longer a Lipschitz matrix, so the general uniqueness theory in [48] is not valid. Nonetheless, such $\mathbf{A}$ are slope matrices, and existence can be verified. Such matrices are appropriate for (at least the initial steps of) interval Newton iteration. The drawback is that the matrix must be recomputed with each new $\check{X}$.

Hansen's technique applied to our example problem with $\check{X} = [0, 0]^T$ and a specified ordering[13] to $\partial\phi/\partial x_2$ gives the matrix:

$$[\text{ Hansen Hessian }] = \begin{bmatrix} [1, 1.25] & [0, 0.25] \\ [0, 0.25] & 1 \end{bmatrix}. \tag{1.9}$$

An alternate point-based way of computing matrices $\mathbf{A}$, more well-known than Hansen's scheme, is utilization of *interval slopes*. Simple one-dimensional examples show that, for $\tilde{\mathbf{X}}$ a point, uniqueness is *not* implied if $\tilde{\mathbf{X}} \subset \text{int}(\mathbf{X})$ when

[12] Other orderings are also possible; cf. [19, §6.4].

[13] cf. [19, §6.3-6.4]. In this example, we are holding x_1 fixed in computing $\mathbf{a}_{2,2}$.

A is formed from interval slopes. However, such **A** are slope matrices in the sense of Definition 2, so existence is implied. Furthermore, as introduced in [58], uniqueness can be verified over a large box **X** by first implying existence over a small box $\check{\mathbf{X}}$ centered at a point $\check{X}$, then showing that the slope matrix over a large box **X** and centered on $\check{\mathbf{X}}$ is regular.

As described by various authors, slope matrices may be computed by operator overloading with techniques similar to automatic differentiation. A sharp scheme, applicable when **X** is large but subject to roundout error for small **X**, appears in [58]. An alternate scheme that can be combined with that of [58] appears in [48]. With the scheme of [58] applied to the exact gradient vector $\nabla\phi = [x_1^3/3 + x_1 + x_2^3/3, x_1x_2^2 + x_2]^T$ and $\check{\mathbf{X}} = \check{X} = [0,0]^T$, the slope matrix for our example is:

$$[\text{Computed slope}] = \begin{bmatrix} [1, 1.0834] & [0, 0.08334] \\ [0, 0.25] & [.75, 1.25] \end{bmatrix}. \tag{1.10}$$

Actually, Hansen's technique and automatic computation of slopes can be combined. In particular, if components of $\check{\mathbf{X}}$ are substituted for components of **X** as in [19, §6.4] and an automatic slope computation scheme is used, the resulting matrix is a slope matrix in the sense of Definition 2. Thus, existence verification, as well as uniqueness verification over large boxes with the two-stage scheme of [58] can proceed with such **A**.

The matrix **A** obtained for our example by combining slope computation with Hansen's idea is:

$$[\text{Hansen–slope}] = \begin{bmatrix} [1, 1.0834] & [0, 0.08334] \\ [0, 0.25] & 1 \end{bmatrix}. \tag{1.11}$$

Based on this present state-of-the-art, we have the following recommendations:

- Hansen's idea combined with slopes should be used:
  - if **A** is to be used only for interval iteration over relatively large boxes.
  - if existence verification only is desired;
  - if uniqueness verification only in as large a box **X** as possible is desired (for example, for deleting subregions in a global search, as in [3]);

  - if non-existence verification only is desired.
  - any combination of the above.

- An interval Jacobi matrix should be used:
  - for interval iteration over small boxes where an interval Jacobi matrix is adequate for convergence, especially if $\check{X}$ must change frequently.
  - if uniqueness verification is required over small boxes $\mathbf{X}$.

If existence only is required over a small box, such as for checking feasibility in Algorithm 3 of §2.1, Hansen's technique combined with slopes is theoretically sharper than interval Hessian matrices. That is, the Hansen-slope scheme is in general preferable in ϵ-inflation algorithms that determine existence. However, depending on the implementation of slopes, the computed slopes may actually be wider than corresponding interval derivatives[14]. Also, other considerations, such as ease of implementation, may enter into consideration.

3.4 Choice of Base Point $\check{X}$

Little appears in the literature concerning selection of the point $\check{X}$. Centered forms for bounding the range, satisfying optimality conditions, were given in [2]. These forms, based on particular choice of point $\check{X}$, can be used when iterating interval Newton methods.

Common choices of $\check{X}$ include the center of the box $\mathbf{X}$, a root of $F(X) = 0$, or both. Some analysis of the case when $\check{X}$ is both a root and the center of the box appears in [63]. This choice, natural in ϵ-inflation when one wishes to verify bounds on a solution obtained through a conventional floating-point method, appears to make verification more likely. An early, abstract analysis of verification tests in [27] indicates that verification is easier if the root of $F(X) = 0$ is near the center of the box. Pictures in [36] also illustrate that it is wise for $\check{X}$ in Algorithm 3 to be a good approximation to a feasible point (corresponding to a root of F), and for the box $\hat{\mathbf{X}}$ in that algorithm to be centered on such a feasible point. Nonetheless, additional analysis of both choosing $\check{X}$ to be a root of F and choosing $\check{X}$ to be centered in the box $\mathbf{X}$ can be done.

[14] The divided differences in [58] should *not* be used for boxes $\mathbf{X}$ that are small in relation to the computational precision.

A graphical examination of the case $F : \mathbb{R}^1 \to \mathbb{R}^1$ suggests that verification of existence or non-existence using slopes can sometimes be more effective when $\check{X}$ is chosen so $\check{X} \notin \mathbf{X}$. Further investigation of this may be useful.

In summary, the present state of knowledge indicates that both roots of $F(X) = 0$ and $\check{X}$ should be centered in $\mathbf{X}$, where possible, if existence or uniqueness verification is desired.

3.5 The Fritz–John Conditions

In certain circumstances, we wish to simultaneously verify feasibility and local optimality. For instance, a conventional floating point constrained optimizer may have been used to obtain an approximate global optimum $\check{X}$. We may then wish to verify uniqueness of a critical point in as large a box $\mathbf{X}$ as possible about $\check{X}$, so that $\mathbf{X}$ may be excluded from further consideration in the exhaustive search[15]. The system of equations to be used in the interval Newton method must therefore embody the necessary conditions for constrained optima. General conditions, not requiring "constraint qualification" assumptions, are the Fritz–John conditions.

The Fritz–John conditions have been advocated by Hansen *et al* in [20] and [19]. Their use is thoroughly explained in those works, although empirical results are lacking. Here, for convenience, we present the function F and derivative matrix ∇F corresponding to the Fritz–John conditions for our formulation in equation (1.1).

The variables in our system are $X = (x_1, \ldots, x_n)$, $V = (v_1, \ldots, v_m)$, and u_0, for a total of $n + m + 1$ variables. We will write $W = (X, u_0, V)$. The function F is:

$$F(W) = \begin{bmatrix} u_0 \nabla \phi(X) + \sum_{i=1}^{m} v_i \nabla c_i(X) \\ c_1(X) \\ \vdots \\ c_m(X) \\ \left[u_0 + \sum_{i=1}^{m} v_i [1, 1+\epsilon]\right] - 1 \end{bmatrix} = 0, \tag{1.12}$$

The v_i are unconstrained and represent Lagrange multipliers for the equality constraints $c_i = 0$. The last equation is a normalization condition suggested and justified in [20]; ϵ is on the order of the computational precision.

[15] This is done in the unconstrained case in [3] and for general nonlinear systems in [35].

By not including the bound constraints $a_{i_j} \le x_{i_j} \le b_{i_j}$ in this function, we reduce the size of the system by $2q$. Furthermore, it is more flexible to include the bound constraints through the process reviewed in §2.3. Thus, equation (1.12) is applicable for points X when none of the bound constraints are active. When one or more bound constraints are active, an analogue of equation (1.12) in the appropriate lower-dimensional subspace is used. In fact, it can be shown that, if the entire space is used and the bound constraints are included in the Fritz–John function F, then the Jacobi matrix of F must be singular; cf. [36] for a precise statement of this fact.

The Jacobi matrix of F, used to form the matrix $\mathbf{A}$ for interval Newton methods, is:

$$H(W) = \left[\begin{array}{c|c|c|c|c} u_0\nabla^2\phi(X) + \sum_{i=1}^{m} v_i\nabla^2 c_i(X) & \nabla\phi(X) & \nabla c_1(X) & \dots & \nabla c_m(X) \\ \hline \nabla c_1(X) & 0 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \nabla c_m(X) & 0 & 0 & \dots & 0 \\ \hline 0 & 1 & [1, 1+\epsilon] & \dots & [1, 1+\epsilon] \end{array}\right]. \tag{1.13}$$

If Hansen's technique as explained in §3.3 is used to evaluate H for the matrix $\mathbf{A}$ in the interval Newton method, then only (guess) point values of u_0 and the v_i enter. Thus, if Gaussian elimination is used to solve equation (1.4), initial bounds on the Lagrange multipliers u_0 and v_i are not required. This procedure, first suggested in [20], is recommended by us, although further experimentation is needed to ascertain what are reasonable guesses for the base points for u_0 and the v_i. In verifying points obtained by floating point constrained optimization software, approximate Lagrange multiplier values may be available along with the approximate optimizers.

4 APPLICATIONS

Actual successful applications of constrained global optimization codes have to date been limited by general lack of availability of such codes to engineers and scientists. A notable exception is the work in [39], in which an algorithm to determine feasibility of constraints is presented and used in structural design analysis with composite laminates. There has also been some success at applying related general nonlinear equation codes or associated techniques. For

example, Schnepper and Stadtherr [59], [60] have modified the TOMS algorithm INTBIS [30] for efficient analysis of chemical process-based flowsheeting. K. Okumura, (private communication), tailoring the techniques to the problem, has applied interval methods for nonlinear systems to analysis of linear and nonlinear electrical networks. In fact, not only did Okumura achieve rigor, but the method proved to be orders of magnitude faster than a Monte Carlo simulation.

In [14], Hager has used techniques in common with interval global optimization algorithms to determine when certain constraints are violated, for use in robot sensing and manipulation, etc.

Csendes and Ratz [6, problem Ex-2, §4] have recently applied Ratz' unconstrained global optimization code successfully to a parameter estimation problem related to respiratory mechanical model identification.

More generally, there has been a steady, and recently increasing interest in interval nonlinear equations technology and interval computations in general for robust geometric computations in computer-aided geometric design. An early review of such techniques is [47]. Computation of the solution of polynomial systems of equations in this context appears in [62]. Experiments with an interval algorithm for intersection of curves appear in [61]. A clever application of interval arithmetic to solution of the nonlinear systems involved in interval ray tracing, allowing substantial speedup of the process, appears in [9].

Our own examination of the realistic test problems in [12] leads us to believe that there is much potential for practical application of interval global optimization algorithms in chemical engineering and process design.

5 SUMMARY AND PRESENT WORK

Various researchers have produced codes for verified global optimization. The books [52] and [19] are devoted to the subject, while Arnold Neumaier is preparing a third. Additionally, the book [48] treats verified solution of nonlinear algebraic systems, while several books are devoted to the underlying interval arithmetic techniques.

While techniques for constrained problems have been published, high-quality experimental reports appear to exist only for unconstrained optimization. Also,

with the possible exception of the codes of [15] or [54], most verified optimization programs to date have been research codes, or have been written in non-portable programming languages.

Besides reviewing this work, we have carefully proposed a method for handling both equality and inequality constraints, and have proposed a variant of a method of Hansen for rigorously verifying feasibility. We have advocated a particular structure when handling inequality, equality, and bound constraints. We have also suggested combining interval slopes with an idea of Hansen for derivative matrices.

We have been developing a portable Fortran-90 environment to support global optimization and numerical nonlinear algebra. The environment, described in [34], is based on the portable FORTRAN-77 package INTLIB of [33]. We intend to complete and test a comprehensive optimization code in this system. This code will be based on the ideas in this work and on the principle that the global algorithm should build on solutions of approximate optimizers that have been verified.

Acknowledgements

I wish to thank George Corliss, Eldon Hansen, Dietmar Ratz, Vladik Kreinovich and other researchers of this subject for their encouragement, careful reading, and suggestions.

REFERENCES

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

[2] Baumann, E., *Optimal Centered Forms*, BIT **28** (1), pp. 80–87, 1988.

[3] Caprani, O., Godthaab, B., and Madsen, K., *Use of a Real-Valued Local Minimum in Parallel Interval Global Optimization*, Interval Computations **1993** (2), pp. 71–82, 1993.

[4] Conn, A. R., Gould, N. and Toint, Ph.L., *LANCELOT: A Fortran Package for Large-Scale Nonlinear Optimization*, Springer-Verlag, New York, 1992.

[5] Csendes, T., *Nonlinear Parameter Estimation by Global Optimization – Efficiency and Reliability*, Acta Cybernetica **8** (4), pp. 361–370, 1988.

[6] Csendes, T. and Ratz, D., *Subdivision Direction Selection in Interval Methods for Global Optimization*, preprint, 1994.

[7] Dennis, J. E., and Schnabel, R. B., *Numerical Methods for Unconstrained Optimization and Nonlinear Least Squares*, Prentice-Hall, Englewood Cliffs, NJ, 1983.

[8] Du, Kaisheng and Kearfott, R. B., *The Cluster Problem in Global Optimization The Univariate Case*, Computing (Suppl.) **9**, pp. 117–127, 1992.

[9] Enger, W., *Interval Ray Tracing – A Divide and Conquer Strategy for Realistic Computer Graphics*, The Visual Computer **9**, pp. 91–104, 1992.

[10] Eriksson, J., *Parallel Global Optimization using Interval Analysis*, Ph.D. dissertation, University of Umeå, Institute of Information Processing, 1991.

[11] Eriksson, J and Lindstrœm, P., *A Parallel Interval Method Implementation for Global Optimization Using Dynamic Load Balancing*, accepted for publication in Interval Computations.

[12] Floudas, C. A. and Pardalos, P. M., *A Collection of Test Problems for Constrained Global Optimization Algorithms*, Springer-Verlag, New York, 1990.

[13] Gill, P. E., Murray, W., and Wright, M., *Practical Optimization*, Academic Press, New York, 1981.

[14] Hager, G. D., *Solving Large Systems of Nonlinear Constraints with Application to Data Modeling*, preprint, 1993.

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

[16] Hanson, Eldon, *Interval Forms of Newton's Method*, Computing **20**, pp. 153–163, 1978.

[17] Hansen, E. R., *Global Optimization Using Interval Analysis: The One-Dimensional Case*, J. Optim. Theory Appl. **29** (3), pp. 331-44, 1979.

[18] Hansen, E. R., *Global Optimization Using Interval Analysis: the Multidimensional Case*, Numer. Math. **34** (3), pp. 247–270, 1980.

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

[20] Hansen, E. R. and Walster, G. W., *Bounds for Lagrange Multipliers and Optimal Points*, Comput. Math. Appl. **25** (10), pp. 59, 1993.

[21] Ichida, K., and Fujii, Y., *An Interval Arithmetic Method for Global Optimization*, Computing **23** (1), pp. 85-97, 1979.

[22] Jansson, C., *A Global Minimization Method: The One-Dimensional Case*, technical report no. 91.2, 1991.

[23] Jansson, C., *A Global Optimization Method Using Interval Arithmetic*, in Computer Arithmetic and Enclosure Methods. Proc. Third International IMACS-GAMM Symposium on Computer Arithmetic and Scientific Computing, ed. L. Atanassova and J. Herzberger, pp. 259–268, North–Holland, Amsterdam, Netherlands, 1992.

[24] Jansson, C. and Knüppel, O., *A Global Minimization Method: The Multi-Dimensional Case*, preprint, 1992.

[25] Jansson, C., *On Self-Validating Methods for Optimization Problems*, in Topics in Validated Computations, ed. J. Herzberger, pp. 381–439, North-Holland, Amsterdam, Netherlands, 1994.

[26] Jansson, C. and Knüppel, O., *Numerical Results for a Self-Validating Global Optimization Method*, technical report, 1994.

[27] Kearfott, R. B., *Abstract Generalized Bisection and a Cost Bound*, Math. Comp. **49** (179), pp. 187–202, 1987.

[28] Kearfott, R. B., *Interval Arithmetic Techniques in the Computational Solution of Nonlinear Systems of Equations: Introduction, Examples, and Comparisons*, in Computational Solution of Nonlinear Systems of Equations (Lectures in Applied Mathematics, volume 26), ed. E. L. Allgower and K. Georg, pp. 337–358, American Mathematical Society, Providence, RI, 1990.

[29] Kearfott, R. B., *Preconditioners for the Interval Gauss–Seidel Method*, SIAM J. Numer. Anal. **27** (3), pp. 804–822, 1990.

[30] Kearfott, R. B., and Novoa, M., *INTBIS, A Portable Interval Newton/Bisection Package (Algorithm 681)*, ACM Trans. Math. Software **16** (2), pp. 152–157, 1990.

[31] Kearfott, R. B., Hu, C. Y., Novoa, M. III, *A Review of Preconditioners for the Interval Gauss–Seidel Method*, Interval Computations **1** (1), pp. 59–85, 1991.

[32] Kearfott, R. B., *An Interval Branch and Bound Algorithm for Bound Constrained Optimization Problems*, Journal of Global Optimization **2**, pp. 259–280, 1992.

[33] Kearfott, R. B., Dawande, M., Du K.-S. and Hu, C.-Y., *Algorithm 737: INTLIB: A Portable FORTRAN 77 Interval Standard Function Library*, ACM Trans. Math. Software **20** (4), pp. 447–459, 1994.

[34] Kearfott, R. B., *A Fortran 90 Environment for Research and Prototyping of Enclosure Algorithms for Constrained and Unconstrained Nonlinear Equations*, ACM Trans. Math. Software **21** (1), pp. 63–78, 1995.

[35] Kearfott, R. B., *Empirical Evaluation of Innovations in Interval Branch and Bound Algorithms for Nonlinear Algebraic Systems*, preprint, 1994.

[36] Kearfott, R. B., *On Verifying Feasibility in Equality Constrained Optimization Problems*, preprint, 1994.

[37] Kearfott, R. B. and Du, K., *The Cluster Problem in Multivariate Global Optimization*, Journal of Global Optimization **5**, pp. 253–265, 1994.

[38] Krawczyk, R., *Newton-Algorithmen zur Bestimmung von Nullstellen mit Fehlershranken*, Computing **4**, pp. 187–201, 1969.

[39] Kristinsdottir, B. P., Zabinsky, Z. B., Csendes, T., Tuttle, M. E., *Methodologies for Tolerance Intervals*, Interval Computations **1993** (3), pp. 133–147, 1993.

[40] Leclerc, A., *Parallel Interval Global Optimization in C++*, Interval Computations **1993** (3), pp. 148–163, 1993.

[41] Mayer, G., *Epsilon–Inflation in Verification Algorithms*, preprint, 1993.

[42] Moore, R. E., *A Test for Existence of Solutions to Nonlinear Systems*, SIAM J. Numer. Anal. **14** (4), pp. 611–615, 1977.

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

[44] Moore, R. E. and Ratschek, H., *Inclusion Functions and Global Optimization II*, Math. Prog. **41** (3), pp. 341–356, 1988.

[45] Moore, R. E., Hansen, E., and Leclerc, A., *Rigorous Methods for Parallel Global Optimization*, in Recent Advances in Global Optimization, ed. A. Floudas and P. Pardalos, pp. 321–342, Princeton Univ. Press, Princeton, N.J., 1992.

[46] Moré, J. J. and Wright, S. J., *Optimization Software Guide*, SIAM, Philadelphia, 1993.

[47] Mudur, S. P. and Koparkar, P. A., *Interval Methods for Processing Geometric Objects*, IEEE Comput. Graphics and Appl. **4** (2), pp. 7–17, 1984.

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

[49] Neumaier, A., *Second-Order Sufficient Optimality Conditions for Local and Global Nonlinear Programming*, preprint, 1994.

[50] Pardalos, P. M., and Rosen, J. B., *Constrained Global Optimization: Algorithms and Applications*, Springer-Verlag, New York, 1987.

[51] Pardalos, P. M. and Vavasis, S. A., *Quadratic Programming with One Negative Eigenvalue is NP-Hard*, Journal of Global Optimization **1** (1), 1992.

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

[53] Ratschek, H., and Voller, R. L., *Global Optimization over Unbounded Domains*, SIAM J. Control Optim. **28** (3), pp. 528–539, 1990.

[54] Ratz, D., *Automatische Ergebnisverifikation bei globalen Optimierungsproblemen*, Ph.D. dissertation, Universität Karlsruhe, 1992.

[55] Ratz, D., *Box-Splitting Strategies for the Interval Gauss–Seidel Step in a Global Optimization Method*, Computing **53**, pp. 337–354, 1994.

[56] Rohn, J., *NP-Hardness Results for Linear Algebraic Problems with Interval Data*, preprint, 1994.

[57] Rump, S. M., *Kleine Fehlerschranken bei Matrixproblemen*, Ph.D. dissertation, Universität Karlsruhe, 1980.

[58] Rump, S. M., *Verification Methods for Dense and Sparse Systems of Equations*, in Topics in Validated Computations, ed. J. Herzberger, pp. 63–135, North–Holland, Amsterdam, 1994.

[59] Schnepper, C. A., *Large Grained Parallelism in Equation-Based Flowsheeting Using Interval Newton / Generalized Bisection Techniques*, Ph.D. dissertation, University of Illinois, Urbana, Department of Chemical Engineering, 1992.

[60] Schnepper, C. A. and Stadtherr, M. A., *Application of a Parallel Interval Newton/Generalized Bisection Algorithm to Equation-Based Chemical Process Flowsheeting*, Interval Computations **1993** (4), pp. 40–64, 1993.

[61] Sederberg, T. W., and Parry, S. R., *Comparison of Three Curve Intersection Algorithms*, Comput. Aided Des. **18** (1), pp. 58–63, 1986.

[62] Sherbrooke, E. C. and Patrikalakis, N. M., *Computation of the Solutions of Nonlinear Polynomial Systems*, preprint, 1993.

[63] Shi, X. and Kearfott, R. B., *Some Results on the Regularity of an Interval Matrix*, preprint, 1994.

[64] Shi, X, *Intermediate Expression Preconditioning and Verification for Rigorous Solution of Nonlinear Problems.* Ph.D. dissertation, University of Southwestern Louisiana, Department of Mathematics, August, 1995.

[65] Skelboe, S., *Computation of Rational Interval Functions*, BIT **14**, pp. 87–95, 1974.

[66] Walster, G. W., Hansen, E. R. and Sengupta, S., *Test Results for a Global Optimization Algorithm*, in Numerical Optimization 1984, ed. P. T. Boggs, R. H. Byrd, and R. B. Schnabel, pp. 272–287, SIAM, Philadelphia, 1985.

[67] Wolfe, M. A., *An Interval Algorithm for Constrained Global Optimization*, J. Comput. Appl. Math. **50**, pp. 605–612, 1994.

INDEX