CONSTRUCTION OF VALIDATED UNIQUENESS REGIONS FOR NONLINEAR PROGRAMS IN WHICH CONVEX SUBSPACES HAVE BEEN IDENTIFIED

R. BAKER KEARFOTT*

Abstract. In deterministic global optimization algorithms for constrained problems, it can be advantageous to identify and utilize coordinates in which the problem is convex, as Epperly and Pistikopoulos have done. In self-validating versions of these algorithms, a useful technique is to construct regions about approximate optima, within which unique local optima are known to exist; these regions are to be as large as possible, for exclusion from the continuing search process. In this paper, we clarify the theory and develop algorithms for constructing such large regions, when we know the problem is convex in some of the variables. In addition, this paper clarifies how one can validate existence and uniqueness of local minima when using the Fritz John equations in the general case. We present numerical results that provide evidence of the efficacy of our techniques.

Key words. nonconvex optimization, global optimization, computational complexity, automatic differentiation, GlobSol, symbolic computation, linear relaxation

AMS subject classifications. 90C30, 65K05, 90C26, 68Q25

1. Introduction.

We consider the general global optimization problem

$$
\begin{array}{ll}
& \text{minimize } \varphi(x) \\
& \text{subject to } c_i(x) = 0,\ i = 1, \ldots, m_1, \\
& \qquad\qquad\ g_i(x) \le 0,\ i = 1, \ldots, m_2, \\
(1.1) & \text{where } \varphi : \boldsymbol{x} \to \mathbb{R} \text{ and } c_i, g_i : \boldsymbol{x} \to \mathbb{R}, \text{ and where } \boldsymbol{x} \subset \mathbb{R}^n \text{ is the} \\
& \text{hyperrectangle (box) defined by} \\
& \qquad \boldsymbol{x} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n) = ([\underline{x}_1, \overline{x}_1], [\underline{x}_2, \overline{x}_2], \ldots, [\underline{x}_n, \overline{x}_n]), \\
& \text{where the } \underline{x}_i \text{ and } \overline{x}_i \text{ are constant bounds.}
\end{array}
$$

Branch and bound algorithms for solving Problem (1.1) proceed by an exhaustive search of the hyperrectangle ("box") $\boldsymbol{x}$, consisting of the following general elements.

(branching) subdividing $\boldsymbol{x}$, forming two sub-boxes $\boldsymbol{x}^{(1)}$ and $\boldsymbol{x}^{(2)}$ by bisecting one of the coordinate intervals, that is, by replacing $[\underline{x}_i, \overline{x}_i]$ by $[\underline{x}_i, (\underline{x}_i + \overline{x}_i)/2]$ and by $[(\underline{x}_i + \overline{x}_i)/2, \overline{x}_i]$, for some i.

(bounding) obtaining lower bounds $\underline{\varphi}(\boldsymbol{x}^{(i)})$ on the objective over the sub-boxes.

(fathoming) rejecting sub-boxes such that $\underline{\varphi}(\boldsymbol{x}^{(i)}) > \overline{\varphi}$, where $\overline{\varphi}$ is an upper bound on the global optimizer[1].

(recursion) further subdividing sub-boxes whose diameters are still large and which aren't fathomed.

In this procedure, the lower bounds on the objective become sharper as the diameters of the sub-boxes become small. Typically, the coordinate i for bisection is chosen to be the one with a scaled maximum width[2]. If the coordinate of maximum width is bisected at each step, then 2^n boxes must, in general, be produced to uniformly subdivide $\boldsymbol{x}$ into boxes that have half the diameter of the original box. Thus, the entire process can be impractical for a large number of variables n.

---

*Department of Mathematics, University of Louisiana, U.L. Box 4-1010, Lafayette, Louisiana, 70504-1010, USA (rbk@louisiana.edu).

[1] The upper bound on the global optimizer is typically obtained by evaluating the objective at feasible points, such as those obtained by local optimization software.

[2] with various scalings, such as the "maximum smear" or "Ratz scaling" of [8, (5.1),p. 175].

In [2], Epperly and Pistikopoulos showed that, if the objective and constraints are convex[3] with respect to $n-k$ of the variables, then only the k non-convex variables need to be subdivided, thus reducing the underlying dimension in the branch and bound process from n to k. Experiments in [2] show a clear advantage for this procedure.

The Epperly/Pistikopoulos procedure relies on the idea that, if Problem (1.1) is convex, then it can be approximated arbitrarily closely by a *linear relaxation*, that is, by a linear program (LP) whose optimum φ^* is less than or equal to the optimum of the original problem (1.1); see [25] for a description of the process. By a "close approximation," we mean that the solution x^{LP} to the optimum of the relaxation is near the solution φ^*. In the convex case, with the addition of slack variables, the objective can be replaced by a linear objective and a convex constraint, and each constraint can be replaced by a set of linear constraints whose feasible region contains the feasible region of the original problem, but whose extent outside the original feasible region is arbitrarily small. In [13, §1], we present a formal view, along with some examples, of an automated process for producing a linear relaxation and determining which variables are convex.

If some of the variables occur in a non-convex way, then the entire problem can be approximated well (as described above) only if the coordinate bounds corresponding to the non-convex coordinates are sufficiently narrow. As the non-convex coordinates are subdivided, various processes other than bisection, such as those described in [25, Ch. 5], can be used to obtain narrower bounds on the convex coordinates. Also, sharp lower bounds $\underline{\varphi}(\boldsymbol{x})$ can be obtained from the linear relaxation, even if these convex coordinates are not small. In these ways, sub-boxes can be fathomed, even though subdivision occurs only in the non-convex variables.

1.1. Validated Branch and Bound. By *validated* methods for solving Problem (1.1), we mean methods that take account of roundoff error in a mathematically rigorous way. Such methods return bounds $\boldsymbol{\varphi}^*$ on the optimum φ^* and bounds $\boldsymbol{x}^*$ on the optimizer x^*, such that, in the absence of programming, operating system, and hardware errors, completion of the algorithm with bounds $\boldsymbol{\varphi}^*$ and $\boldsymbol{x}^*$ is a mathematical proof that $\varphi^* \in \boldsymbol{\varphi}^*$ and $x^* \in \boldsymbol{x}^*$.

Work by Neumaier and Shcherbina [20] and independently by Jansson [7] has enabled validated solution of Problem (1.1) with linear relaxations. Several researchers have developed validated algorithms employing validated linear relaxations, including [1], [16], us ([10]), and others.

We have applied the linear relaxations in a validated context in [13]. However, although our basic algorithmic structure is similar to that in [2], we did not observe the advantage of subdivision in a subspace that was apparent in [2]. This could be due to the fact that we have not yet devised validated versions of all of the narrowing techniques in [25, Ch. 5]. Moreover, additional techniques, such as interval Newton methods and ϵ-inflation with box complementation have been used in validated, but not non-validated algorithms to exclude parts of the search region; in our work in [13], we did not structure these techniques to take advantage of convexity.

In this paper, we first point out and derive relevant aspects of convexity analysis in §2. Next, in §3, we review and derive results for the interval Newton procedures

[3]We say that an inequality constraint $g(x) \le 0$ is convex if and only if g is a convex function, while we say an equality constraint $c(x) = 0$ is convex if and only if both c and $-c$ are convex functions

to be used. We present the overall goals, as well as details, of the algorithms in §4, while numerical experiments appear in §5. We summarize in §6.

2. Convex Subspaces. Here, we clarify our terminology and present a theorem underlying our validation procedure that makes use of convex variables.

DEFINITION 2.1. *Consider Problem (1.1), consider a set of indices* $\mathcal{P} \subset \{i\}_{i=1}^n$ *with cardinality* n_p*, and fix* $\xi_i \in \boldsymbol{x}_i$ *for* $i \in \mathcal{P}$*. For each such choice, define a new problem* $(1.1)_{\mathcal{P}}$ *of the same form as Problem (1.1) but with* φ*,* c*, and* g *replaced by* $\tilde{\varphi}$*,* $\tilde{c}$*, and* $\tilde{g}$*, where* $\tilde{\varphi} : \mathbb{R}^{n-n_p} \to \mathbb{R}$*,* $\tilde{c} : \mathbb{R}^{n-n_p} \to \mathbb{R}^{m_1}$*, and* $\tilde{g} : \mathbb{R}^{n-n_p} \to \mathbb{R}^{m_2}$ *are defined by fixing* x_i *at* ξ_i *in* φ*,* c*, and* g*, respectively, for* $i \in \mathcal{P}$*. We say that Problem* $(1.1)_{\mathcal{P}}$ *is the* derived subspace problem *to Problem (1.1) at* $(\xi_{i_1}, \ldots, \xi_{i_{n_p}})$*,* $i_j \in \mathcal{P}$*.*

DEFINITION 2.2. *We say that Problem (1.1) is convex with respect to the variable set* $\mathcal{Q} \subseteq \{i\}_{i=1}^n$*, provided the derived subspace problem* $(1.1)_{\mathcal{P}}$ *is convex for each choice of the* $\xi_i \in \boldsymbol{x}_i$*,* $i \in \mathcal{P}$*, where* $\mathcal{P} = \{i\}_{i=1}^n \setminus \mathcal{Q}$*.*

The following theorem underlies one of our important algorithm constructs.

THEOREM 2.3. *Assume*

1. $x^\dagger$ *is a local optimizer of Problem (1.1).*
2. $x^\dagger$ *is known to be unique within a set of bounds* $\boldsymbol{x}^\dagger = (\boldsymbol{x}_1^\dagger, \ldots, \boldsymbol{x}_n^\dagger)$*,* $\boldsymbol{x}_i^\dagger \subseteq \boldsymbol{x}_i$ *for* $i \in \{i\}_{i=1}^n$*.*
3. *Problem (1.1) is convex with respect to a variable set* $\mathcal{Q}$*.*
4. *Define* $\mathcal{P} = \{i\}_{i=1}^n \setminus \mathcal{Q}$*, and partition the coordinates of* $x \in \boldsymbol{x}$ *into* $x_{\mathcal{P}} = (x_{i_1}, \ldots, x_{i_{n_p}})$*,* $i_j \in \mathcal{P}$ *and* $x_{\mathcal{Q}} = (x_{i_1}, \ldots, x_{i_{n-n_p}})$*,* $i_j \in \mathcal{Q}$*; assume* $(1.1)_{\mathcal{P}}$ *has a unique solution* $x_{\mathcal{Q}} \in \boldsymbol{x}_{\mathcal{Q}}^\dagger$ *at each point* $x_{\mathcal{P}} \in \boldsymbol{x}_{\mathcal{P}}^\dagger$*.*

Then, the local optimizer is has the smallest objective value of any point x *within the new box* $\tilde{\boldsymbol{x}}^\dagger$ *obtained by replacing* $\boldsymbol{x}_i^\dagger$ *by the corresponding entire coordinate range* $\boldsymbol{x}_i$*, for each* $i \in \mathcal{Q}$*.*

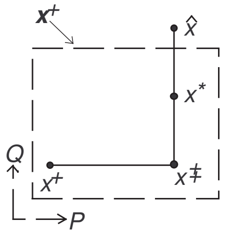

FIG. 2.1. *Construction in the proof of Theorem 2.3.*

Proof. Suppose there is a feasible point $\hat{x} \in \tilde{\boldsymbol{x}}^\dagger$ for which $\varphi(\hat{x}) \leq \varphi(x^\dagger)$, and form an intermediate point $x^\ddagger = (x_{\mathcal{P}}^\ddagger, x_{\mathcal{Q}}^\ddagger)$ such that $x_{\mathcal{P}}^\ddagger = \hat{x}_{\mathcal{P}}$ and $x_{\mathcal{Q}}^\ddagger = x_{\mathcal{Q}}^\dagger$. (See Figure 2.1.) Then, Assumption 3 implies that there is an $x_{\mathcal{Q}}^* \in \boldsymbol{x}_{\mathcal{Q}}^\dagger$ that is the unique solution to $(1.1)_{\mathcal{P}}$ for parameters $x_{\mathcal{P}}^\ddagger$. Assumption 3 then implies that φ is nondecreasing on the line segment between $(\hat{x}_{\mathcal{P}}, x_{\mathcal{Q}}^*)$ and $\hat{x}$, that is, $\varphi(\hat{x}_{\mathcal{P}}, x_{\mathcal{Q}}^*) \leq \varphi(\hat{x})$. Furthermore, Assumptions 1 and 2 and the inference that $(\hat{x}_{\mathcal{P}}, x_{\mathcal{Q}}^*)$ must be feasible (as a solution to $(1.1)_{\mathcal{P}}$) implies that $\varphi(x^\dagger) = \varphi(x_{\mathcal{P}}^\dagger, x_{\mathcal{Q}}^*) < \varphi(x^*) = \varphi(\hat{x}_{\mathcal{P}}, x_{\mathcal{Q}}^*)$, whence $\varphi(x^\dagger) < \varphi(\hat{x})$, contradicting the assumption on $\hat{x}$. Therefore, $\varphi(x) > \varphi(x^\dagger)$ for every $x \in \tilde{\boldsymbol{x}}^\dagger$, and the theorem is proved. □

Assumption 3 is checked in a symbolic preprocessing step, as explained in [2] and

[13], while Assumptions 1, 2, and 4 can be checked in a natural way with the interval Gauss–Seidel method, as we explain below.

3. Interval Newton Methods, Convex Subspaces, and Constraints. We review and develop properties of interval Newton methods in the context of validating the assumptions of Theorem 2.3 here.

3.1. Interval Newton Methods. The underlying theory goes back to early work of Moore, Nickel, and others, and is based on classical fixed point theory. A general theorem is [18, Theorem 5.1.7], and we review this theory in [8, §1.5].

Here, we use upper case letters X to denote vectors which we will view as being in a higher-dimensional space that includes the Lagrange multipliers for Problem (1.1) as components, in addition to the components of the variables x from Problem (1.1).

Suppose $F : \mathbb{R}^M \to \mathbb{R}^M$, and let $\mathbb{IR}^M$ denote the set of interval M-vectors. Then a general form for multivariate interval Newton methods is

$$\tilde{\boldsymbol{X}} = \mathbf{N}(F, \boldsymbol{X}, \check{X}) = \check{X} + \boldsymbol{V}, \tag{3.1}$$

where $\boldsymbol{V} \in \mathbb{IR}^M$ contains all solutions V to point systems $AV = -F(\check{X})$, for $A \in \boldsymbol{F}'(\boldsymbol{X})$, where $\boldsymbol{F}'(\boldsymbol{X})$ is an interval extension to the Jacobi matrix of F over $\boldsymbol{X}$, or is an interval matrix containing a slope set for F over $\boldsymbol{X}$. Then, under conditions such as those in [18, Theorem 5.1.7] and [8, §1.5],

1. $\mathbf{N}(F, \boldsymbol{X}, \check{X})$ must contain all solutions $X^* \in \boldsymbol{X}$ with $F(X^*) = 0$;
2. If $\mathbf{N}(F, \boldsymbol{X}, \check{X})$ is contained in the interior of $\boldsymbol{X}$, then there is a solution of $F(X) = 0$ within $\mathbf{N}(F, \boldsymbol{X}, \check{X})$, and hence within $\boldsymbol{X}$.
3. If, in addition $\boldsymbol{F}'(\boldsymbol{X})$ also is an interval derivative matrix, or if the technique in [24] is used, and $\mathbf{N}(F, \boldsymbol{X}, \check{X})$ is contained in the interior of $\boldsymbol{X}$, then the solution to $F(X) = 0$ in $\boldsymbol{X}$ is unique.

A particular interval Newton method, the interval Gauss–Seidel method has various desirable general properties; in addition, its componentwise nature is appropriate for our context of validation in subspaces. Suppose $\boldsymbol{A}$ represents the interval extension to the Jacobi matrix (or interval slope matrix), and $\boldsymbol{a}_{i,j}$ is the entry in the i-th row, j-th column of $\boldsymbol{A}$. Then the image under the interval Gauss–Seidel method is defined by

$$\tilde{\boldsymbol{X}}_i = \check{X}_i - \left(F_i(\check{X}) + \sum_{j=1}^{i-1} \boldsymbol{a}_{i,j}(\tilde{\boldsymbol{X}}_j - \check{X}_j) + \sum_{j=i+1}^{M} \boldsymbol{a}_{i,j}(\boldsymbol{X}_j - \check{X}_j) \right) \Big/ \boldsymbol{a}_{i,i}, \tag{3.2}$$

where $\tilde{\boldsymbol{X}}_i$ is defined sequentially for $i = 1$ to M. Also, $F(\check{X})$ is usually replaced by $YF(\check{X})$ and $\boldsymbol{A}$ is usually replaced by $Y\boldsymbol{A}$, where Y is a preconditioner matrix; see [8, Ch. 3] for a development of such preconditioners[4]. We say we have completed a "sweep" of the interval Gauss–Seidel method provided we have sequentially computed $\tilde{\boldsymbol{X}}_i$, $i = 1, 2, \ldots, n$. We may iterate the interval Gauss–Seidel method through one or more sweeps. We have the following validation properties.

THEOREM 3.1. *Suppose $F : \mathbb{R}^M \to \mathbb{R}^M$, and suppose we obtain $\tilde{\boldsymbol{X}}$ from one or more sweeps of the interval Gauss–Seidel method (3.2), and $\tilde{\boldsymbol{X}}_i$ is contained in the interior of $\boldsymbol{X}_i$ for each $i \in \{i\}_{i=1}^M$. Then:*

[4]In the context here, where we are assuming the coordinates of $\boldsymbol{X}$ are relatively narrow, an appropriate preconditioner is the "inverse midpoint matrix," that is, an approximate inverse of the matrix consisting of the midpoints of the elements of $\boldsymbol{A}$.

1. *If a slope matrix is used for the matrix $\boldsymbol{A}$, then, regardless of whether or not preconditioning was used in the iteration formula (3.2), there is a solution to the original system $F(X) = 0$ within $\tilde{\boldsymbol{X}}$, and hence within $\boldsymbol{X}$.*
2. *If, in addition, $\boldsymbol{A}$ is also an interval derivative matrix, then, regardless of whether or not preconditioning was used in the iteration formula (3.2), there is a unique solution to the original system $F(X) = 0$ within $\tilde{\boldsymbol{X}}$, and hence within $\boldsymbol{X}$.*

Theorem 3.1, well known (ibid.), allows us to use the interval Gauss–Seidel method to verify Assumptions 1 and 2 of Theorem 2.3.

The following theorem, of particular use in the subspace context here, follows from Theorem 3.1 and fundamental properties of interval arithmetic.

THEOREM 3.2. *Suppose that $F : \mathbb{R}^M \to \mathbb{R}^{M-n_p}$, $\mathcal{Q} = \{i_j\}_{j=1}^{M-n_p} \subseteq \{i\}_{i=1}^{M}$, and the interval Gauss–Seidel iteration (3.2) is applied, starting with $\boldsymbol{X}$, to only coordinates $i_j \in \mathcal{Q}$ and not to coordinates $i_j \in \mathcal{P} = \{i\}_{i=1}^{M} \setminus \mathcal{Q}$. Suppose that the image under this process has $\tilde{\boldsymbol{X}}_{i_j}$ in the interior of $\boldsymbol{X}_{i_j}$ for all $i_j \in \mathcal{Q}$, and partition the coordinates of X into $X_{\mathcal{P}}$ and $X_{\mathcal{Q}}$ as in the proof of Theorem 2.3. Then, for each $X_{\mathcal{P}} \in \boldsymbol{X}_{\mathcal{P}}$, there is a solution $X_{\mathcal{Q}} \in \boldsymbol{X}_{\mathcal{Q}}$ such that $F(X) = F(X_{\mathcal{P}}, X_{\mathcal{Q}}) = 0$.*

Combined with the considerations below regarding the Fritz John system, Theorem 3.2 allows us to use the interval Gauss–Seidel method to verify Assumption 4 for Theorem 2.3.

3.2. The Fritz John and Kuhn–Tucker Systems. As has been suggested in [4] and elsewhere, we have used the well-known Fritz John conditions in our GlobSol [9, 14] software for our F in the interval Newton validation process for Problem (1.1). Specifically, let $u = (u_0, u_1, \ldots, u_{m_2})$ represent the multipliers for φ and the g_i, let $v = (v_1, \ldots, v_{m_1})$ represent the multipliers for the c_i, let $M = n + m_1 + m_2 + 1$, and define $X \in \mathbb{R}^M$ by $X = (x, u, v)$. The Fritz John conditions then become

$$F(X) = \begin{pmatrix} u_0 \nabla\varphi(x) + \sum_{1=1}^{m_2} u_i \nabla g_i(x) + \sum_{i=1}^{m_1} v_i \nabla c_i(x) \\ u_1 g_1(x) \\ \vdots \\ u_{m_2} g_{m_2}(x) \\ c_1(x) \\ \vdots \\ c_{m_1}(x) \\ \left(u_0 + \sum_{i=1}^{m_2} u_j + \sum_{i=1}^{m_1} v_i^2\right) - 1 \end{pmatrix} = 0, \tag{3.3}$$

With the normalization represented by the last equation, bounds on u and v are $u_i \in [0, 1]$, $0 \le i \le m_2$ and $v_i \in [-1, 1]$, $1 \le i \le m_1$.

Applying the interval Gauss–Seidel method as described in §3 to the Fritz John equations $F(X) = 0$ of (3.3) proves there is a unique critical point, but not necessarily a local minimizer, of Problem (1.1). A sufficient condition for such a critical point to be a local minimizer is for the projected Hessian matrix of the Lagrangian function to be positive definite, that is, for the projected Jacobi matrix with respect to the variables x of

$$u_0 \nabla \varphi(x) + \sum_{1=1}^{m_2} u_i \nabla g_i(x) + \sum_{i=1}^{m_1} v_i \nabla c_i(x) \tag{3.4}$$

to be positive definite (and $u_0 \neq 0$); see [3, §3.4] for a discussion of this, and see §3.3 below.

Suppose Problem (1.1) is convex with respect to the variable set $\mathcal{Q}$, suppose $\mathcal{P} = \{i\}_{i=1}^{n} \setminus \mathcal{Q}$, and suppose $X^\dagger$ has been proven via Theorem 3.1 to be a unique critical point of the Fritz John system (3.3) within $\boldsymbol{X}^\dagger$. If we verify that the projected Hessian of the Lagrangian function is positive definite at the critical point $x^\dagger$, then this verifies Assumption 1 of Theorem 2.3. Furthermore, suppose Theorem 3.2 has also been used to prove existence of a critical point of Problem $(1.1)_{\mathcal{P}}$ for each $X_{\mathcal{P}} \in \boldsymbol{X}_{\mathcal{P}}^\dagger$; this combined with convexity with respect to $\mathcal{Q}$ then implies Assumption 4 of Theorem 2.3. This is how our computational validation will actually proceed.

In our context here, we have used the Kuhn–Tucker system instead of the Fritz John system. In the Kuhn Tucker system, the multipliers are not normalized, and u_0 is fixed at 1 and removed from the variable set; thus, the Kuhn–Tucker system consists of (3.3) with the last equation removed and u_0 replaced by the constant 1. The Kuhn–Tucker system is slightly more common in the literature, but the Fritz John system handles some exceptional cases not handled by the Kuhn–Tucker system. In validated branch and bound algorithms, we have preferred the Fritz John system in GlobSol because a priori bounds on the multipliers ($[0, 1]$ for the u's and $[-1, 1]$ for the v's) are available. However, if the context is validation and we already have a high-quality approximation to the Lagrange multipliers u and v corresponding to the Kuhn–Tucker equations, we can construct small intervals about these approximate values, and it is more convenient to use the Kuhn Tucker equations rather than try to convert the Lagrange multipliers to multipliers for the Fritz John equations.

3.3. Null Spaces of Constraints. Suppose x^* is an exact solution to the Fritz John system (3.3), and suppose we have identified the set of active inequality constraints, that is, suppose we have identified that subset $\left\{g_{i_j}\right\}_{j=1}^{m_a}$, $m_a \leq m_2$ for which $g_{i_j}(x^*) = 0$. (The remaining inequality constraints have $g_i(x^*) < 0$, and have corresponding multipliers $u_i = 0$.) Denote by H the Jacobi matrix of the Fritz John gradient (3.4) (or Kuhn–Tucker system) with respect to the primal variables x, let $\{G_{i,:}\}_{i=1}^{m_1+m_a}$ denote the exact gradients of the active inequality and equality constraints at x^*. Select a maximal linearly independent subset $\mathcal{G}$ of these so that, without loss of generality, we may assume $m = m_1 + m_a \leq n$, and denote by $G \in \mathbb{R}^{m \times n}$ the matrix whose i-th row is $G_{i,:} \in \mathcal{G}$. If $m = n$ and the constraint gradients are linearly independent, then the projected Hessian matrix is 0, and x^* must correspond to a local minimum. Otherwise, let $\{Z_{:,i}\}_{i=1}^{n-m}$ be a basis for the null space of G, and let $Z \in \mathbb{R}^{n-m \times n}$ be that matrix whose i-th column is $Z_{:,i}$, so the projected Hessian matrix is

$$H_P = Z^T J_x Z, \tag{3.5}$$

where J_x is the Jacobi matrix with respect to the variables x of the expression (3.4), and we need to verify that H_P is positive definite.

To take account of roundoff errors to mathematically rigorously show H_P is positive definite, is suffices to have

1. proof that the chosen set $\mathcal{G}$ of constraint gradients is linearly independent, that is, proof that the matrix G is of full rank m (needed for subsequent validations);
2. validated bounds on the components of J_x;
3. validated bounds on the components of the columns of Z;
4. proof that the set of constraint gradients chosen for the matrix G spans the space spanned by all active constraints; and
5. proof that the matrix H_P, formed with the rigorous bounds on J_x and Z, is positive definite.

Since the exact solution $X^\dagger$ to the Fritz John equations (3.3) is not known precisely, but since iteration of the interval Newton method will give us a small box $\boldsymbol{X}^\dagger$ within which there will exist a unique such solution, we will need to verify the above properties over the box $\boldsymbol{X}^\dagger$ (whose x-coordinates are $\boldsymbol{x}^\dagger$), rather than at a computed approximation to $X^\dagger$. We now explain how to verify each of the above properties in this context.

3.3.1. Verification of linear independence. Verification that G is of full rank will proceed simply by

ALGORITHM 1. *(verifying that $G \in \mathbb{R}^{m\times n}$, $m \le n$, is of rank m)*
INPUT: the narrow bounds $\boldsymbol{x}^\dagger$ corresponding to the x-coordinates of the critical point and an interval evaluation $G(\boldsymbol{x}^\dagger)$ corresponding to the ranges of the constraint gradients over $\boldsymbol{x}^\dagger$.
OUTPUT: either "rank validated" or "rank not validated".

1. *Compute an approximate Moore–Penrose pseudo-inverse Y of the matrix of midpoints of $\boldsymbol{G}^T(\boldsymbol{x}^\dagger)$ using common floating-point arithmetic.*
2. *Form $\Upsilon = Y\boldsymbol{G}^T(\boldsymbol{x}^\dagger)$ using interval arithmetic.*
3. IF $\inf \Upsilon_{i,i} > \sum_{j=1, j\ne i}^{m} \sup |\Upsilon_{i,j}|$ *for each i between 1 and m,*
   THEN RETURN *"rank validated"*
   ELSE RETURN *"rank not validated".*

END ALGORITHM 1

3.3.2. Validated Bounds on the Components of the Kuhn–Tucker Jacobi Matrix. Computing validated bounds on the components of J_x can be done simply by evaluating $J_x(\boldsymbol{X}^\dagger)$ using interval arithmetic.

3.3.3. Validated Bounds on the Basis Vectors for the Null Space of the Constraints. Provided the widths of $\boldsymbol{X}^\dagger$ are sufficiently small, computing validated bounds on the components of Z can be done as we have explained in [12] for point matrices, as follows.

ALGORITHM 2. *(Compute bounds on a null space basis for the constraints.)*
INPUT: An interval matrix $\boldsymbol{G}(\boldsymbol{x}^\dagger) \in \mathbb{IR}^{m\times n}$ whose rows are interval evaluations of the gradients of the active constraints over $\boldsymbol{x}^\dagger$.
OUTPUT: Either (i) a set of interval vectors $\boldsymbol{Z}_{:,i}$ $1 \le i \le n-m$, such that, for each $G \in \boldsymbol{G}(\boldsymbol{x}^\dagger)$, there is a $Z_{:,i} \in \boldsymbol{Z}_{:,i}$ for $1 \le i \le n-m$ with $\{Z_{:,i}\}_{i=1}^{n-m}$ forming an orthonormal basis for the null space of G, or (ii) "not validated".

1. *Find the midpoint matrix $\check{G}$ of $\boldsymbol{G}(\boldsymbol{x}^\dagger)$.*
2. *As in step 1 of the method in [12], find an approximate basis $\check{Z} = [\check{Z}_{:,1}, \ldots, \check{Z}_{:,n-m}]$ to the null space of $\check{G}$, using a traditional method, such as an LU or QR factorization or a singular value decomposition.*
3. *Construct a small box $\boldsymbol{Z}$ about $\check{Z}$ as in step 2 of the method in [12]. However, the widths of the box should be constructed in this case to take account of the*

non-zero widths of $\boldsymbol{G}(\boldsymbol{x}^\dagger)$.

4. *As in step 3 of the method in [12], apply an interval Newton method to the $n(n-m)$ by $n(n-m)$ system $\boldsymbol{G}(\boldsymbol{x}^\dagger)Z = 0$, $Z^T Z = I$, with starting box $\boldsymbol{Z}$ and base point $\check{Z}$.*
5. *If the previous step has resulted in proof that a basis Z exists in $\boldsymbol{Z}$, then iterate the interval Newton method to obtain bounds $\boldsymbol{Z}$ that are as small as possible for the matrix $\boldsymbol{G}(\boldsymbol{x}^\dagger)$.*

END ALGORITHM 2

The method of [12] requires that each matrix in $G(\boldsymbol{x}^\dagger)$ be of full rank, which will happen if the constraint gradients are continuous, usual interval extensions are used, and the constraint gradients are linearly independent at the solution $x^\dagger$. However, if M constraints have been identified as possibly active at the solution in $\boldsymbol{x}^\dagger$, and it is suspected (by computations with a floating point algorithm) that the gradients of these constraints are linearly dependent at the solution, it is in general not possible to prove that a linearly independent set of $m \le n$ of them spans the space of active constraints, unless $m = n$. To see this, suppose it happens that $n = 2$, suppose that two constraints are active at the solution, and suppose the gradients of these constraints appear to be both approximately $(1, 0)$. A floating point algorithm will give evidence that, to within a specified tolerance, the two computed approximations to the gradients are nearly equal to $(1, 0)$. However, even if the approximately computed gradients are both exactly equal to $(1, 0)$, a standard interval technique will begin by constructing boxes $\boldsymbol{g}^{(1)} = ([1-\epsilon_1, 1+\epsilon_1], [-\epsilon_2, \epsilon_2])$ and $\boldsymbol{g}^{(2)} = ([1-\epsilon_3, 1+\epsilon_3], [-\epsilon_4, \epsilon_4])$ to take account of possible errors in the floating point computations. It is clear that any two such boxes with non-zero widths will contain vectors that are not linearly dependent. Thus, our algorithms will fail in this case. (Linear dependence of this type probably needs to be detected symbolically, in preprocessing steps.)

3.3.4. Verifying that the Selected Set of Active Constraint Gradients Spans the Space Spanned by the Gradients of All Active Constraints. This is done by simply selecting all “possibly active” constraints. That is, we evaluate each component $g_i(\boldsymbol{x}^\dagger)$ using interval arithmetic, discarding only those g_i for which $g_i(\boldsymbol{x}^\dagger) < 0$ from the list of active constraints. This guarantees that no active constraints are overlooked.

3.3.5. Verifying that the Projected Hessian Matrix is Positive Definite. Once we obtain bounds $\boldsymbol{J}_x(\boldsymbol{X}^\dagger)$ for the Jacobi matrix of (3.4) and once we obtain a matrix $\boldsymbol{Z} \in \mathbb{IR}^{n\times(n-m)}$ whose columns bound a basis for the null space of the active constraints at $\boldsymbol{x}^\dagger$, we may form a bound

$$\boldsymbol{H}_P = \boldsymbol{Z}^T \boldsymbol{J}_x(\boldsymbol{X}^\dagger)\boldsymbol{Z} \tag{3.6}$$

for the projected Hessian matrix. The exact matrix, contained within $\boldsymbol{H}_P$, is symmetric, so it is sufficient to prove that all symmetric matrices in $\boldsymbol{H}_P$ are positive definite. For this purpose, we can use the following theorem due to Rohn and Neumaier.

THEOREM 3.3. *([23, 17, 19]) Suppose $\boldsymbol{H}$ is a symmetric interval matrix, and suppose there is a symmetric matrix $H_0 \in \boldsymbol{H}$ that is positive definite. If, in addition, each symmetric matrix $H \in \boldsymbol{H}$ is non-singular, it follows that each symmetric matrix $H \in \boldsymbol{H}$ is positive definite.*

With Theorem 3.3, the following algorithm will verify that the projected Hessian matrix is positive definite.

ALGORITHM 3. *(Showing that the projected Hessian matrix at the solution is positive definite)*

INPUT: the matrix $\boldsymbol{H}_P$ formed as in (3.6).
OUTPUT: either "positive definiteness verified" or "positive definiteness not verified."

1. *Compute a symmetric floating point approximation $\check{H}_P$ to the matrix of midpoints of elements of $\boldsymbol{H}_P$, making sure $\check{H}_P$ is symmetric and $\check{H}_P \in \boldsymbol{H}_P$.*
2. *Use a floating point routine to compute approximate eigenvalues $\lambda_1 \le \lambda_2 \ldots \le \lambda_{n-m}$ and corresponding approximate orthonormal eigenvectors $\{U_{:,i}\}_{i=1}^{n-m}$ of $\check{H}$.*
3. IF $\lambda_1 \le 0$ THEN RETURN *"positive definiteness not verified."*
4. *Construct small intervals $\boldsymbol{\lambda}_i$ centered on each λ_i and small boxes $\boldsymbol{U}_{:,i}$ centered on each $U_{:,i}$.*
5. *Apply an interval Newton method $n-m$ times to the system $\check{H}u - \lambda u = 0$, $u^T u = 1$ of $n-m+1$ equations in $n-m+1$ unknowns[5], with base point $(\lambda_i, U_{:,i})$ and initial box $(\boldsymbol{\lambda}_i, \boldsymbol{U}_{:,i})$, $1 \le i \le m-n$ to provide verified bounds on each of the λ_i.*
6. IF *the interval Newton method in step 5 fails to validate bounds on any of the λ_i* THEN RETURN *"positive definiteness not validated".*
7. IF *the smallest lower bound on any of the λ_i is non-positive,* THEN RETURN *"positive definiteness not validated".*
8. *Compute an approximate inverse Y to $\check{H}_P$, then form $\boldsymbol{\Upsilon} = Y\boldsymbol{H}_P$.*
9. IF $\inf \Upsilon_{i,i} > \sum_{j=1, j\ne i}^{n-m} \sup |\Upsilon_{i,j}|$ *for each i between 1 and $n-m$,*
   THEN RETURN *"positive definiteness validated"*
   ELSE RETURN *"positive definiteness not validated".*

END ALGORITHM 3

3.4. Slope Matrices. We use the technique first appearing in [6], and later recommended in [5] (and see [8, §1.3.2] for a review) to form the slope matrices. In particular, in evaluating the slope matrices for (3.3), the values for the components of u and v can be points, except in the last row (the normalization equation).

When slope matrices are used instead of interval derivative matrices, an interval Newton method can prove existence, but not uniqueness. However, a technique in [24] can be used to prove uniqueness in a larger box. In particular, let $\mathbf{S}(F, \boldsymbol{X}, \check{\boldsymbol{X}})$ represent a slope matrix[6] for F over the box $\boldsymbol{X}$ with respect to points $\check{X} \in \check{\boldsymbol{X}}$, and let $\check{X} \in \check{\boldsymbol{X}}$. Then:

1. if the hypotheses of Theorem 3.1 hold for $\boldsymbol{A} = \mathbf{S}(F, \check{\boldsymbol{X}}, \check{X})$, then there exists a solution of $F(X) = 0$ in $\check{\boldsymbol{X}}$;
2. if, in addition, the hypotheses of Theorem 3.1 hold for $\boldsymbol{A} = \mathbf{S}(F, \boldsymbol{X}, \check{\boldsymbol{X}})$ for some $\boldsymbol{X}$ with $\check{\boldsymbol{X}} \subseteq \boldsymbol{X}$, then the solution of $F(X) = 0$ in $\check{\boldsymbol{X}}$ is unique within $\boldsymbol{X}$.

4. The Overall Algorithms.

4.1. Overall Algorithm Goals. The basic idea is to prove uniqueness of a critical point in the entire space within a box with widths of the non-convex coordinates that are as large as possible, but with the widths of the convex coordinates set small (to avoid overestimation with interval dependency), but large enough to make it likely that their corresponding image under the Gauss–Seidel operator is inside their original bounds. For this latter condition to be likely, the widths of the convex coordinates should be scaled according to the widths of the non-convex coordinates.

[5] This system has appeared at various places in the literature, probably first in [15]
[6] A slope matrix is a matrix that contains a slope set in the sense defined on [18, p. 202].

Once a box is found in which uniqueness can be verified, convexity implies that the solution is unique within a box in which the convex coordinates have an extent that is arbitrarily wide.

Our algorithms result in:

1. a small box $\boldsymbol{x}^*$ within which a critical point of the Fritz John system is proven to exist,
2. a large box $\boldsymbol{x}^{\square}$, $\boldsymbol{x}^* \subseteq \boldsymbol{x}^{\square}$, such that there is a unique local minimizer for Problem (1.1) within $\boldsymbol{x}^{\square}$.

4.2. The Main Algorithm. The following algorithm uses Theorem 3.1 to construct a box satisfying Assumptions 1 and 2 of Theorem 2.3.

ALGORITHM 4. *(Scaling the convex coordinates, return a small box in which existence of a critical point to the Kuhn–Tucker system for Problem (1.1) is proven and a larger box in which uniqueness of that critical point is proven.)*

INPUT:

α. a current sub-box $\boldsymbol{x}$,

β. a list $\mathcal{P} = \{i_j\}_{j=1}^{n_p}$, $1 \le i_j \le n$ of "parameter coordinates," such that Problem (1.1) is convex with respect to the variable set $\mathcal{Q} = \{i\}_{i=1}^{n} \setminus \mathcal{P}$.

OUTPUT:

α. Either "existence validated" or "existence not validated." If "existence validated," then also output a small box $\boldsymbol{x}^* \in \mathbb{IR}^n$, in which there exists a solution to the Kuhn–Tucker equations.

β. Either "uniqueness validated" or "uniqueness not validated." If "uniqueness validated," then also output a large box $\boldsymbol{x}^{\square} \in \mathbb{IR}^n$, within which uniqueness is proven.

1. *Compute the midpoint $\hat{x}$ of $\boldsymbol{x}$.*
2. *Use a state-of-the-art nonlinear programming solver with starting point $\hat{x}$ to compute an approximate local optimizer $\check{X}$ to Problem (1.1), as well as corresponding Lagrange multipliers $\check{u}$ and $\check{v}$ for the constraints, that is, to return $\check{X} = (\check{x}, \check{u}, \check{v})$.*
3. IF *the optimization was unsuccessful or if interval evaluations of the constraints over a small box constructed about $\check{x}$ certify that one or more constraints is infeasible over that box* THEN RETURN *with "existence not verified".*
4. *(Verify existence with an interval Newton method)*
   (a) *Construct a small box $\boldsymbol{X}^{\square}$ around $\check{X}$ with fixed, small coordinate widths ϵ.*
   (b) *Apply an interval Newton method (Algorithm 5 below), with base "box" and base point both equal to $\check{X}$, and containing box equal to $\boldsymbol{X}^{\square}$ and parameter set empty, to verify existence of a solution to the Kuhn–Tucker equations.*
   (c) IF *Algorithm 5 returned "not validated"*
   THEN
   RETURN *"existence not validated".*
   ELSE
   (Produce as small box a box as possible within which a solution exists.)
   DO
   i. *Execute Algorithm 5 with containing box $\boldsymbol{X}^{\dagger}$, base box and base point both equal to the midpoint of $\boldsymbol{X}^{\dagger}$, and parameter set empty.*

ii. IF *Algorithm 5 returns "false" for "progress"* THEN EXIT DO
iii. $\boldsymbol{X}^\dagger \leftarrow \boldsymbol{X}^*$, *where* $\boldsymbol{X}^*$ *is the image returned by Algorithm 5.*
END DO
END IF

(d) *Identify candidate active constraints at* $X^\dagger$, *as explained in §3.3.4.*
IF *the number of possibly active constraints exceeds* n THEN RETURN *with "existence not validated".*

(e) *Use Algorithm 1, where* $G(\boldsymbol{x}^\dagger)$ *is the matrix whose* i*-th column is an interval enclosure for the gradient of the* i*-th candidate active constraint over* $\boldsymbol{x}^\dagger$.
IF *linear independence could not be validated* THEN RETURN *with "existence not validated".*

(f) *Use Algorithm 2 to compute bounds on the vectors* $\boldsymbol{Z}$ *in the nullspace of* $G(\boldsymbol{x}^\dagger)$.

(g) *Use Formula 3.6 to compute an interval enclosure* $\boldsymbol{H}_P(\boldsymbol{X}^\dagger)$ *for the projected Hessian matrix over* $\boldsymbol{X}^\dagger$

(h) *Use Algorithm 3 to verify that the projected Hessian matrix* $\boldsymbol{H}_P$ *is positive definite.*
IF *positive definiteness could not be validated*
THEN
RETURN *with "existence not validated".*
ELSE
Set $\boldsymbol{x}^*$ *to be the* x*-coordinates of* $\boldsymbol{X}^\dagger$, *and set "existence validated".*
END IF

5. *(Compute as large a box as possible within which a solution is unique.)*
(a) i. $\boldsymbol{x}_i^{\Box} \leftarrow \boldsymbol{X}_i^\dagger$ *for* $1 \le i \le n$;
ii. $\boldsymbol{X}^{\square} \leftarrow \boldsymbol{X}^\dagger$.
(b) $\iota \leftarrow 1$.
(c) *(Inflate the non-convex coordinates.)*
DO
i. *For* ϵ *the same as in the existence step 4a, set*
A. $\underline{\underline{X}}_i^{\square} \leftarrow \underline{\underline{X}}_i^{\square} - 2^\iota\epsilon$ *for* $1 \le i \le M$.
B. $\overline{\overline{X}}_i^{\square} \leftarrow \overline{\overline{X}}_i^{\square} + 2^\iota\epsilon$ *for* $1 \le i \le M$.
ii. *Execute Algorithm 5 with base box* $\boldsymbol{X}^\dagger$, *base point* $X^\dagger$ *equal to the midpoint of* $\boldsymbol{X}^\dagger$, *and containing box* $\boldsymbol{X}^{\square}$.
iii. IF *Algorithm 5 returned*[7] *"validated"*
THEN
A. *"uniqueness validated"* $\leftarrow$ *"true".*
B. $\boldsymbol{x}_i^{\Box} \leftarrow \boldsymbol{X}_i^{\square}$ *for* $1 \le i \le n$.
ELSE
A. IF $\iota = 1$ THEN *"uniqueness validated"* $\leftarrow$ *"false".*
B. EXIT*step 5c.*
END IF
iv. $\boldsymbol{X}^{\Box} \leftarrow \boldsymbol{X}^{\square}$; $\boldsymbol{x}^{\Box} \leftarrow (\boldsymbol{X}_1^{\Box}, \ldots, \boldsymbol{X}_n^{\Box})$.
END DO
(d) $\iota \leftarrow \iota + 1$.

6. *(Verify assumption 4 of Theorem 2.3)*

[7]This validates the uniqueness part of Assumption 2 of Theorem 2.3.

(a) *verified* $\leftarrow$ *"false".*

(b) DO WHILE *(not verified)*

i. *Execute Algorithm 5 with base box and containing box both equal to* $\boldsymbol{X}^{\square}$, *base point equal to the midpoint of* $\boldsymbol{X}^{\dagger}$, *and parameter set* $\mathcal{P}$ *equal to the non-convex coordinates.*

ii. IF *Algorithm 5 returned "validated",* THEN

A. *Set* $\boldsymbol{x}_i^{\square} \leftarrow \boldsymbol{x}_i$ *for* $i \in \mathcal{Q} = \{i\}_{i=1}^{n} \setminus \mathcal{P}$.

B. RETURN

ELSE

A. *Replace* $\boldsymbol{X}_i^{\square}$ *by an interval with the same midpoint and half its width, for* $i \in \mathcal{P}$.

B. IF $\boldsymbol{X}_i^{\dagger} \not\subset \boldsymbol{X}_i^{\square}$ *for any* $i \in \mathcal{P}$ THEN

RETURN

END IF

END IF

END DO

END ALGORITHM 4

NOTE 1. *If step 5 of Algorithm 4 completes with "uniqueness validated" "true", then the large box* $\boldsymbol{X}^{\square}$ *contains a unique local optimum of Problem (1.1) that, furthermore, must lie within the smaller box* $\boldsymbol{X}^{\dagger}$. *Algorithm 4 is likely to complete this step. The last step, step 6 of Algorithm 4, is to verify that the convex coordinates can be replaced by their entire extents. Because assumption 4 of Theorem 2.3 is not general, this step may not complete. However, even if step 6 does not complete, Algorithm 4 will still return a box in which uniqueness has been proven, albeit one in which the parameter coordinates not be very wide.*

As explained in [24], the following algorithm can prove existence within $\boldsymbol{X}^{\square}$ when the "center" box $\boldsymbol{X}^{\dagger}$ is a point, and uniqueness within both $\boldsymbol{X}^{\dagger}$ and $\boldsymbol{X}^{\square} \supseteq \boldsymbol{X}^{\dagger}$ when $\boldsymbol{X}^{\dagger}$ has coordinates with nonzero widths, provided existence has already been proven within $\boldsymbol{X}^{\dagger}$

ALGORITHM 5. *(Our interval Newton method)*

INPUT:

α. a base point (or base box) $\boldsymbol{X}^{\dagger}$, a point $X^{\dagger} \in \boldsymbol{X}^{\dagger}$, a containing box $\boldsymbol{X}^{\square}$, and a set $\mathcal{P}$ of parameter coordinates.

β. a tolerance ϵ for determining whether or not the interval Gauss–Seidel method reduced the relative widths of any of the coordinates.

OUTPUT:

α. Either "validated" or "not validated".

β. "progress" if the relative width of at least one coordinate, when intersected with its original value, is decreased by at least ϵ, and "no progress" if none of the relative widths are so decreased.

γ. An image box $\boldsymbol{X}^*$.

1. *(Compute a slope enclosure.)*

(a) *Evaluate a slope extension* $\mathbf{S}(F, \boldsymbol{X}^{\square}, \boldsymbol{X}^{\dagger})$ *to the Kuhn–Tucker system for Problem (1.1), "centered" at the box* $\boldsymbol{X}^{\dagger}$. *To obtain a slope extension appropriate for validation for Problem* $(1.1)_{\mathcal{P}}$, *replace* $\mathbf{S}(F, \boldsymbol{X}^{\square}, \boldsymbol{X}^{\dagger})$ *by* $\tilde{\mathbf{S}}(F, \boldsymbol{X}^{\square}, \boldsymbol{X}^{\dagger})$ *by ignoring rows (but not columns) of* $\mathbf{S}(F, \boldsymbol{X}^{\square}, \boldsymbol{X}^{\dagger})$ *corresponding to indices in* $\mathcal{P}$.

(b) *Evaluate an approximate inverse* Y *of the midpoint matrix of the matrix obtained from* $\tilde{\mathbf{S}}(F, \boldsymbol{X}^{\square}, \boldsymbol{X}^{\dagger})$ *by ignoring the columns of* $\tilde{\mathbf{S}}(F, \boldsymbol{X}^{\square}, \boldsymbol{X}^{\dagger})$

corresponding to indices in $\mathcal{P}$.

(c) *Compute the Kuhn–Tucker residuals $F(\check{X})$ for Problem (1.1); obtain the Kuhn–Tucker residuals $\tilde{F}(\check{X})$ for Problem $(1.1)_{\mathcal{P}}$ by ignoring rows corresponding to partial derivatives with respect to coordinates whose indices are in $\mathcal{P}$.*

2. *(Validate existence or uniqueness)*
   (a) *Apply the interval Gauss–Seidel method to the preconditioned system $Y\tilde{\mathbf{S}}(F, \boldsymbol{X}^{\square}, \boldsymbol{X}^{\dagger})(\boldsymbol{X}^{\square} - X^{\dagger}) = -Y\tilde{F}(X^{\dagger})$ to obtain images $\tilde{\boldsymbol{X}}_i$, $1 \le i \le M$, $i \notin \mathcal{P}$. (Note that all M coordinates appear in $\boldsymbol{X}^{\square}$ and $X^{\dagger}$, but not in the image $\tilde{\boldsymbol{X}}$ under this step.)*
   (b) IF $\tilde{\boldsymbol{X}}_i \not\subset \boldsymbol{X}_i^{\square}$ *for some i between 1 and M,*
   THEN *"validated" ← "false".*
   ELSE *"validated" ← "true".*
   END IF
   (c) *Set the coordinates of $\boldsymbol{X}^*$ to be the intersections of corresponding coordinates of $\tilde{\boldsymbol{X}}$ and $\boldsymbol{X}^{\square}$.*

END ALGORITHM 5

5. Numerical Experiments.

5.1. The Computational Environment. We implemented Algorithm 4 within our GlobSol [9] environment, on a dual 3.2GHz Pentium-4 based machine with 2 gigabytes of memory, running Microsoft Windows XP, and using the Compaq Visual Fortran compiler version 6.6, with optimization level 0. We implemented Algorithm 4 both in a "stand-alone" mode and embedded in our GlobSol algorithm. We used a version of the solver IPOPT [26] from early 2003 to compute the approximate local optimum in step 2 of Algorithm 4.

5.2. Embedding within GlobSol. In addition to trying Algorithm 4 in isolation, we embedded it into our overall branch and bound algorithm (GlobSol) to determine a possible impact on efficiency.

5.2.1. GlobSol's Algorithm. We have published GlobSol's overall algorithm in [9] and [10]. We repeat the algorithm here for clarity.

ALGORITHM 6. *(GlobSol's global search algorithm, [9, 10])*

INPUT: A list $\mathcal{L}$ of boxes $\boldsymbol{x}$ to be searched.

OUTPUT: A list $\mathcal{U}$ of small boxes and a list $\mathcal{C}$ of boxes verified to contain feasible points, such that any global mimimizer must lie in a box in $\mathcal{U}$ or $\mathcal{C}$.

DO WHILE *($\mathcal{L}$ is non-empty)*

1. *Remove a box $\boldsymbol{x}$ from the list $\mathcal{L}$.*
2. IF *$\boldsymbol{x}$ is sufficiently small* THEN
   (a) *Analyze $\boldsymbol{x}$ to validate feasible points, possibly widening the coordinate widths (ϵ-inflation) to a wider box within which uniqueness of a critical point can be proven.*
   (b) *Place $\boldsymbol{x}$ on either $\mathcal{U}$ or $\mathcal{C}$.*
   (c) *Apply the complementation process of [8, p. 154].*
   (d) CYCLE
   END IF
3. *(Constraint Propagation)*
   (a) *Use constraint propagation to possibly narrow the coordinate widths of the box $\boldsymbol{x}$.*

   (b) IF *constraint propagation has shown that $\boldsymbol{x}$ cannot contain solutions* THEN CYCLE
4. *Compute a linear relaxation of the problem (as in [10]) to obtain possibly sharper lower bounds on the global optimum and to possibly reduce the widths of the coordinate vectors or eliminate $\boldsymbol{x}$ totally from the search.*
5. *(Interval Newton)*
   (a) Perform an interval Newton method to possibly narrow the coordinate widths of the box $\boldsymbol{x}$.
   (b) IF *the interval Newton method has shown that $\boldsymbol{x}$ cannot contain solutions* THEN CYCLE
6. IF *the coordinate widths of $\boldsymbol{x}$ are now sufficiently narrow* THEN
   (a) Analyze $\boldsymbol{x}$ to validate feasible points, possibly widening the coordinate widths (ϵ-inflation) to a wider box within which uniqueness of a critical point can be proven.
   (b) Place $\boldsymbol{x}$ on either $\mathcal{U}$ or $\mathcal{C}$.
   (c) Apply the complementation process of [8, p. 154].
   (d) CYCLE
7. *(Subdivide)*
   (a) Choose a coordinate index k to bisect (i.e. to replace $[\underline{x}_k, \overline{x}_k]$ by $[\underline{x}_k, (\underline{x}_k + \overline{x}_k)/2]$ and $[(\underline{x}_k + \overline{x}_k)/2, \overline{x}_k]$).
   (b) Bisect $\boldsymbol{x}$ along its k-th coordinate, forming two new boxes; place these boxes on the list $\mathcal{L}$.
   (c) CYCLE

END DO END ALGORITHM 6

In the actual software, the list $\mathcal{L}$ input to Algorithm 6 is obtained as follows:

1. Begin with an initial box $\boldsymbol{x}$.
2. Using a generalized Newton method based on a pseudo inverse and using the midpoint of $\boldsymbol{x}$ as a starting point, obtain an approximate feasible point $\hat{x}$.
3. Construct a small box $\hat{\boldsymbol{x}}$ about $\hat{x}$.
4. Using, say, techniques from [11], prove existence of an actual feasible point within $\hat{\boldsymbol{x}}$.
5. Take the complement of $\hat{\boldsymbol{x}}$ in $\boldsymbol{x}$ to form the list $\mathcal{L}$ input to Algorithm 6.

We take the complement of a box in a box using a modification of Algorithm 10, [8, p. 155] in which the coordinates are ordered in order of decreasing width. Similarly, we use Algorithm 11 (p. 156, ibid.) to take the complement of a box in a list.

5.2.2. Modifications to GlobSol. We inserted a call to Algorithm 4 between steps 6 and 7 of GlobSol's overall search algorithm (Algorithm 6), passing the current box $\boldsymbol{x}$ and the list $\mathcal{Q}$ obtained from forming the LP relaxations, and took the following actions after return:

IF "existence validated" and if $\boldsymbol{y} \in \mathcal{C}$ then $\boldsymbol{x}^* \notin \boldsymbol{y}$,
THEN

1. Evaluate the objective φ over $\boldsymbol{x}^*$ to possibly obtain a lower upper bound on the global optimum of Problem 1.1 to use in rejecting subregions for which the lower bound on φ is higher.
2. *IF* "uniqueness validated"
   THEN $\boldsymbol{y} \leftarrow \boldsymbol{x}^{\square}$
   ELSE $\boldsymbol{y} \leftarrow \boldsymbol{x}^*$
   END IF
3. Form a list $\mathcal{T}$ by taking the complement of $\boldsymbol{x}^{\square}$ in $\boldsymbol{x}$.

4. Replace $\mathcal{L}$ by the complement of $\boldsymbol{y}$ in $\mathcal{L}$.
5. Replace $\mathcal{U}$ by the complement of $\boldsymbol{y}$ in $\mathcal{U}$.
6. Append to $\mathcal{U}$ the complement of $\boldsymbol{y}$ in $\mathcal{C}$.
7. Insert the boxes in $\mathcal{T}$ into $\mathcal{L}$.
8. Take the first box from $\mathcal{L}$, put it into $\boldsymbol{x}$, and continue GlobSol's overall algorithm.

END IF

We note that, if existence is validated but uniqueness is not, then $\boldsymbol{x}^*$ can have very small widths, and the complementation process may be ineffective. However, we have observed that it is rare for uniqueness not to be validated when existence is.

5.3. The Test Problems. We have used essentially the same problems we employed in [13], [10] and [11]. In particular, our initial test problem, originally appearing in [22] and used as a test problem in [27], is

$$\begin{array}{ll} & \min_{x\in\mathbb{R}^n} x_5 \\ (5.1) & \text{such that} \left\{ \begin{array}{rcl} x_4 - \left(x_1 t_i^2 + x_2 t_i + x_3\right)^2 - \sqrt{t_i} \quad - x_5 & \leq & 0 \\ -\left\{x_4 - \left(x_1 t_i^2 + x_2 t_i + x_3\right)^2 - \sqrt{t_i}\right\} - x_5 & \leq & 0 \end{array} \right\}, \\ & 1 \leq i \leq m, \text{ where } t_i = 0.25 + 0.75(i-1)/(m-1). \end{array}$$

This problem represents a minimax fit to $\sqrt{t}$ with a function of the form $p(t) = x_4 - \left(x_1 t_i^2 + x_2 t_i + x_3\right)^2$, with m equally spaced data points in $[0.25, 1]$.

We tried this problem with $m = 5$ and $m = 21$. We used $\epsilon = 10^{-9}$ in step 4a and step 5c of Algorithm 4 for $m = 5$, and we used $\epsilon = 10^{-7}$ for the $m = 21$ case.

We also tried Example 2 from [2]. We will not give the details of this example here, although computer code to evaluate this example is available from the authors upon request.

Finally, as in [13], [10], and [11], we tried those problems from the Neumaier–Shcherbina "Tiny-1" test set [21] which were translated into the GlobSol format properly, for which GlobSol had supporting routines, and for which the approximate solver (IPOPT) did not produce exceptions that stopped processing. In all of these problems, we used $\epsilon = 10^{-7}$ in step 4a and step 5c of Algorithm 4. The exact way in which the problems were posed to the global optimizer can significantly affect the solution process; source files completely defining the way these problems are posed[8] are available from the author upon request.

When Algorithm 4 was embedded in the GlobSol algorithm (Algorithm 6), in all cases ϵ was set to 10 times the parameter `EPS_DOMAIN` used to determine when a box in GlobSol is too small to bisect further.

5.4. Results in Isolation. We first tried Algorithm 4 in isolation on Problem (5.1) with $m = 5$, with initial box $\boldsymbol{x} = ([0,5],[0,5],[0,5],[0,5],[0,100])$, and with non-convex coordinates (automatically computed) $\mathcal{P} = \{1,2,3\}$. The approximate solver IPOPT immediately computed the solution

$$x \approx [-.08573, .4953, -1.118, 1.502, .002459]$$

and Lagrange multipliers[9]

$$\lambda \approx [.1395, 0, 0, .3037, 0, 0, .3605, 0, 0, .1963]$$

[8] These source files were generated automatically through processes mentioned in [21].

[9] These are readily translated into Fritz John multipliers for our algorithm by ignoring the ones that are zero, then normalizing.

in step 2 of Algorithm 4. Existence of a solution to the Kuhn–Tucker equations was easily validated in step 4b, and step 4c narrowed the solution box to a box contained in

$$\begin{array}{l}
\big([-0.0\underline{87531574373}5, -0.0\underline{87531574373}3],\\
\;[\;\;0.\underline{4953160762}506, \;\;\; 0.\underline{4953160762}510],\\
\;[-\underline{1.118352080853}7, -\underline{1.118352080852}9],\\
\;[\;\;\underline{1.502446927353}2, \;\;\; \underline{1.502446927355}5],\\
\;[\;\;0.00\underline{24593569376}02, \;\;\; 0.00\underline{24593569376}06]\big).
\end{array}$$

The remaining validation sub-steps of step 4 completed successfully, then the inflation process in step 5c proceeded up to $\iota = 17$ successful inflations. Validation of Assumption 4 of Theorem 2.3 (step 6 of Algorithm 4) then completed successfully, and Algorithm 4 returned a box[10]

$$\big([-0.08760, -0.08746], [0.4952, 0.4954], [-1.119, -1.118], [0, 5], [0, 100]\big)$$

in which the solution was proven to be unique.

Results with $m = 21$, as with $m = 5$, all validation steps proceeded successfully, starting with the same initial box and non-convex coordinates as with $m = 5$. However, only five inflation steps (rather than 17 as in the case of $m = 5$) were possible; the final solution box was

$$\begin{array}{l}
\big([-0.0\underline{8801551466 8}97, -0.0\underline{88015514668}84],\\
\;[\;\;0.\underline{4954443098 4}77, \;\;\; 0.\underline{49544430984}81],\\
\;[-\underline{1.1186219560 5}17, -\underline{1.11862195605}09],\\
\;[\;\;\underline{1.5031597385 7}32, \;\;\; \underline{1.50315973857}50],\\
\;[\;\;0.00\underline{263597349736}70, \;\;\; 0.00\underline{263597349736}95]\big),
\end{array}$$

and the inflated box (in which the solution was proven to be unique), rounded out in the last digit we display here, was

$$\big([-.088019, -.088012], [0.495441, 0.495448], [-1.11863, -1.11861], [0, 5], [0, 100]\big).$$

When we applied Algorithm 4 to the Epperly–Pistikopoulos Example 2 with initial box

$$\begin{array}{llllllll}
\big([5, 10], & [5, 10], & [5, 10], & [0, 180], & [0, 80], & [0, 200], & [0, 80], & [0, 220],\\
\;[0, 80], & [0, 250], & [0, 80], & [0, 180], & [0, 100], & [0, 200], & [0, 100], & [0, 220],\\
\;[0, 100], & [0, 250], & [0, 100], & [0, 180], & [0, 120], & [0, 200], & [0, 130], & [0, 220],\\
\;[0, 130], & [0, 250], & [0, 130], & [0, 180], & [0, 150], & [0, 200], & [0, 150], & [0, 230],\\
\;[0, 150], & [0, 250], & [0, 150], & [3, 8]\big), & & & &
\end{array}$$

IPOPT immediately computed a correct approximate solution in step 2 of Algorithm 4, but the existence validation step, step 4b failed. In fact, the midpoint matrix Y^{-1} in the interval Newton method (Algorithm 5) had a condition number on the order of 10^{17}, singular to working precision. This leads us to believe that, for this problem, the Fritz John system is singular at the solutions, even when inactive constraints are removed from the system, and the algorithms in this paper in their present form are inapplicable.

[10] here, for a more comprehensible display, we have rounded the actual end points out to four significant figures.

5.5. Results within GlobSol. We ran GlobSol's overall search algorithm, modified as indicated in §5.2.2 above, on problem (5.1) ("OET5") with starting box $\boldsymbol{x} = ([-5,5],[-5,5],[-5,5],[-5,5],[-100,100])$, with both $m = 5$ and $m = 21$. For both $m = 5$ and $m = 21$, uniqueness-containing boxes with the two solutions to the problem within $\boldsymbol{x}$ were obtained immediately when GlobSol (Algorithm 6) passed Algorithm 4 the top two boxes in its list $\mathcal{L}$. For example, in the case of $m = 21$, we obtained the following boxes (with coordinates rounded out to 4 digits) in which uniqueness of a local optimum is proven:

$$\big([\ \ 0.08800,\ \ \ 0.08803],[-0.4955,-0.4953],[1.118,1.120],[-5,5],[-100,100]\big)$$
and
$$\big([-0.08803,-0.08800],[\ \ 0.4953,\ \ \ 0.4955],[1.118,1.120],[-5,5],[-100,100]\big).$$

Each subsequent call to Algorithm 4 within GlobSol yielded a successful construction, but all of these constructions corresponded to one of the initial two boxes. In these initial experiments, tried two variants: The first variant involved bisecting (step 7 of Algorithm 6) in all coordinates, and the second involved bisecting only in the non-convex coordinates $\mathcal{P}$, as we did in [13]. For both $m = 5$ and $m = 21$, the final output list $\mathcal{C}$ of boxes with verified feasible points consisted precisely of the two narrow solution boxes, and the list $\mathcal{U}$ of small unresolved boxes was empty. (That is, GlobSol was completely successful in obtaining narrow boxes containing all solutions.)

Efficiency results for Problem 5.1 appear in Table 5.1. In both tables in this section, the column labelled "n" gives the number of variables, "n_r gives the number of non-convex variables, m_1 gives the number of equality constraints, and m_2 gives the number of inequality constraints. The column labelled "V?" has "T" if if Algorithm 4 was used in GlobSol's overall algorithm, and has "F" if Algorithm 4 was not used. The column labelled "B?" has "T" if only the non-convex coordinates were chosen for bisection in GlobSol's overall algorithm (i.e. in step 7 of Algorithm 6), and has "F" if GlobSol was allowed to choose any coordinate (convex or non-convex) for bisection. The column labelled "# boxes" gives the total number of boxes processed in Algorithm 6. The column labelled "CPU" gives the total processor time Algorithm 6 used to terminate. The column labelled "CPU_V" is the total processor time taken in Algorithm 4. (The processor times are rounded to the nearest second.) The final column, labelled "OK?" has "0" if the overall algorithm completed within the specified time and total number of box limits (20,000 seconds for "OET5" and 1 hour per problem for each problem from the Neumaier–Shcherbina test set), and "−1" if the algorithm did not complete within those limits. As seen in Table 5.1, the

TABLE 5.1
Results for Problem 5.1 (OET5) with and without subspace analysis and Algorithm 4

n	n_r	m_1	m_2	V?	B?	# boxes	CPU	CPU_V	OK?
5	—	0	10	F	F	20541	1261	0	0
5	3	0	10	T	F	10812	664	223	0
5	3	0	10	T	T	4779	395	103	0
5	—	0	42	F	F	15688	20000	0	-1
5	3	0	42	T	T	5749	2767	192	0

experiments with $m = 5$ hinted that it is more efficient to allow GlobSol to select only coordinates with indices from the set $\mathcal{P}$ of non-convex coordinates to bisect. For this reason, in all subsequent experiments, we bisected only in the non-convex coordinates whenever we used Algorithm 4. (Bisecting only in the convex coordinated, but not using Algorithm 4 does not seem to be effective, from experiments we reported in [10].)

In Table 5.2, we see that the technique successfully validated optima in 21 out of 32 problems. (The columns sub-labelled "y" correspond to results with Algorithm 4, and the columns labelled "n" correspond to results without Algorithm 4.) Furthermore, we

Table 5.2
Results from the Neumaier–Shcherbina test set with and without Algorithm 4

Problem	n	n_r	m_1	m_2	# boxes		CPU		CPU_V	N_V	OK?	
					y	n	y	n			y	n
dispatch	4	3	1	1	19	18	0.6	0.3	0.2	1	0	0
ex14.1.1	3	2	0	4	119	59	2.6	1.6	0.3	0	0	0
ex14.1.2	6	3	1	8	66840	33073	3601.4	3601.4	877.8	0	-1	-1
ex14.1.5	6	4	4	2	164	77	2.3	3.0	0.0	1	0	0
ex14.1.9	2	1	0	2	30	41	0.6	0.4	0.2	3	0	0
ex14.2.2	4	3	1	4	505	522	22.7	17.0	2.8	0	0	0
ex14.2.5	4	3	1	4	476	493	27.9	17.7	3.0	0	0	0
ex2.1.2	6	5	0	2	168	168	2.5	1.9	0.0	0	0	0
ex3.1.3	6	6	0	6	291	291	66.3	1.9	63.9	0	0	0
ex3.1.4	3	3	0	3	35	35	2.0	0.6	1.1	0	0	0
ex4.1.2	1	1	0	0	8	79	0.8	4.0	0.3	1	0	0
ex4.1.4	1	1	0	0	13	120	6.2	0.1	6.2	1	0	0
ex4.1.5	2	2	0	0	50	148	0.5	0.4	0.3	12	0	0
ex4.1.6	1	1	0	0	5	168	0.0	0.2	0.0	2	0	0
ex4.1.7	1	1	0	0	4	56	0.0	0.1	0.0	2	0	0
ex4.1.8	2	1	1	0	4	9	0.0	0.0	0.0	1	0	0
ex4.1.9	2	1	0	2	27	55	0.4	0.4	0.1	2	0	0
ex5.4.2	8	5	0	6	8110	1155	3612.1	108.7	2916.9	0	-1	0
ex7.2.6	3	3	0	1	58	58	39.8	0.4	39.2	0	0	0
ex7.3.1	4	3	0	7	33	62	0.4	4.1	0.0	1	0	0
ex7.3.2	4	2	0	7	12	12	0.1	0.2	0.0	1	0	0
ex7.3.3	5	1	2	6	23	41	0.4	3.2	0.1	1	0	0
ex8.1.3	2	2	0	0	6948	173362	3600.3	3600.1	438.8	4491	-1	-1
ex8.1.4	2	2	0	0	48	64	0.5	0.2	0.2	14	0	0
ex8.1.5	2	2	0	0	144	282	8.3	1.0	7.3	30	0	0
ex8.1.6	2	2	0	0	9	56	0.3	0.5	0.1	2	0	0
ex9.2.4	8	4	7	0	37802	87840	3656.0	3731.8	2233.9	2	-1	-1
ex9.2.5	8	5	7	0	26361	81590	3642.7	3633.2	2038.5	4	-1	-1
ex9.2.8	3	0	2	0	8	8	0.1	0.1	0.0	0	0	0
mhw4d	5	3	3	0	166	333	7.1	17.1	1.8	6	0	0
rbrock	2	1	0	0	2	28	0.0	0.1	0.0	1	0	0
31					148482	380303	18305	14751	8633	4579	-5	-4

see that the technique reduced the total number of boxes processed for most problems, but the reduction in overall number of boxes processed is hard to interpret because these numbers are highly weighted towards the problems that could not finish in an hour of processor time. Reductions in numbers of boxes processed occurred even for those problems, such as `ex8.1.5` and `ex8.1.6`, that do not have convex subspaces. The total processing time was more when Algorithm 4 was used, and this can be attributed totally to the time spent in the validation process. This is due to the fact that Algorithm 4 is invoked every time a box has not been fathomed in the branch and bound algorithm. However, we have observed that, typically, all of the local optima are found and validated within the first few boxes processed in the branch and bound, and subsequent invocations repeat the process unnecessarily. Good heuristics could thus avoid much of this excess computation.

For the problems that finished, the total number of boxes processed was more with Algorithm 4 than without for problems `ex14.1.1`, `ex14.1.5`, and `ex5.4.2`. (The latter is the only problem that did not finish with Algorithm 4 but finished without it.) This can be explained for these problems from the fact that, when Algorithm 4 was used for this test set, we allowed GlobSol to bisect only in the convex coordinates, which can be less efficient when it is difficult to extend regions to be eliminated in the direction of the convex coordinates (such as when Algorithm 4 is not functioning well).

Algorithm 4 was unable to validate a global optimum for eleven of these problems. We list the reasons for failure in Table 5.3.

TABLE 5.3
Failures to validate with Algorithm 4

Problem	Step in which Failed	Reason / Comments
ex14.1.1	4(b) (Existence of a solution to the K/T system)	The K/T system is apparently singular at the solution.
ex14.1.2	4(b) (Existence of a solution to the K/T system)	The K/T system is singular at the solution, possibly because of linearly dependent constraints there.
ex14.2.2	4(b) (Existence of a solution to the K/T system)	The K/T system is singular at the solution, possibly because of linearly dependent constraints there.
ex14.2.5	4(b) (Existence of a solution to the K/T system)	The K/T system is singular at the solution, possibly because of linearly dependent constraints there.
ex2.1.2	Validation process was never called.	
ex3.1.3	4(a)	The approximate solver (IPOPT) was never successful.
ex4.1.2	4(b) (Existence of a solution to the K/T system)	The K/T system is singular at the solution, possibly because of linearly dependent constraints there.
ex5.4.2	4(a)	The approximate solver (IPOPT) was never successful.
ex7.2.6	4(a)	The approximate solver (IPOPT) was never successful.
ex9.2.8	Validation process was never called.	

6. Conclusions. Condition 4 of Theorem 2.3 is strong (not satisfied, say, for constrained systems where there are n active constraints at the local optimum) and more difficult to verify in practice than the other items. However, the procedures in this paper can be used, without expansion of the convex coordinates, in the general case, assuming there are no convex coordinates. To our knowledge, this is the first time these techniques have been carefully analyzed together, to provide rigorous automatic validation of local optima when approximate optima are computed as stationary points of the Kuhn–Tucker system.

The numerical results provide evidence that the procedure can be practical when astutely woven into a branch and bound algorithm. The procedure can lead to higher efficiency in validated branch and bound algorithms. In any case, the procedures lead to higher-quality answers, in the sense that the answers are narrow boxes that have been proven to contain local optima (rather than boxes with no such proof).

REFERENCES

[1] G. Borradaile and P. Van Hentenryck. Safe and tight linear estimators for global optimization. In *Workshop on Interval Analysis and Constraint Propagation for Applications (IntCP 2003)*, 2003. `http://www.cs.brown.edu/people/pvh/linest.ps`.
[2] T. G. W. Epperly and E. N. Pistikopoulos. A reduced space branch and bound algorithm for global optimization. *J. Global Optim.*, 11(3):287–311, October 1997.
[3] P. E. Gill, W. Murray, and M. Wright. *Practical Optimization*. Academic Press, New York, 1981.
[4] E. R. Hansen. *Global Optimization Using Interval Analysis*. Marcel Dekker, Inc., New York, 1992.
[5] E. R. Hansen and G. W. Walster. Bounds for Lagrange multipliers and optimal points. *Comput. Math. Appl.*, 25(10):59, 1993.
[6] Eldon Hansen. Interval forms of Newton's method. *Computing*, 20(2):153–163, 1978.
[7] Ch. Jansson. A rigorous lower bound for the optimal value of convex optimization problems. *J. Global Optim.*, 28(1):121–137, January 2004.

[8] R. B. Kearfott. *Rigorous Global Search: Continuous Problems.* Kluwer, Dordrecht, Netherlands, 1996.

[9] R. B. Kearfott. Globsol: History, composition, and advice on use. In *Global Optimization and Constraint Satisfaction*, Lecture Notes in Computer Science, pages 17–31, New York, 2003. Springer-Verlag.

[10] R. B. Kearfott. Empirical comparisons of linear relaxations and alternate techniques in validated deterministic global optimization, 2004. accepted for publication in *Optimization Methods and Software.*

[11] R. B. Kearfott. Improved and simplified validation of feasible points: Inequality and equality constrained problems, 2005. preprint,.

[12] R. B. Kearfott. Validated bounds on basis vectors for the null space of a full rank rectangular matrix, 2005. preprint,.

[13] R. B. Kearfott and S. Hongthong. Validated linear relaxations and preprocessing: Some experiments, 2003. accepted for publication in *SIAM J. Optim.*

[14] R. B. Kearfott, M. Neher, S. Oishi, and F. Rico. Libraries, tools, and interactive systems for verified computations: Four case studies, 2003. preprint,.

[15] R. Krawczyk. Error estimates for real eigenvalues and eigenvectors of matrices. *Computing*, 4(4):281–93, 1969.

[16] Y. Lebbah, C. Michel, M. Rueher, D. Daney, and J.-P. Merlet. Efficient and safe global constraints for handling numerical constraint systems. *SIAM J. Numer. Anal.*, 42(5):2076–2097, 2004.

[17] A. Neumaeri. Second-order sufficient optimality conditions for local and global nonlinear programming. *J. Global Optim.*, 9:141–151, 1996.

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

[19] A. Neumaier. Complete search in continuous global optimization and constraint satisfaction. *Acta Numerica*, 2004.

[20] A. Neumaier and O. Shcherbina. Safe bounds in linear and mixed-integer programming. *Math. Prog.*, 99(2):283–296, March 2004.

[21] A. Neumaier, O. Shcherbina, W. Huyer, and T. Vinkó. A comparison of complete global optimization solvers. *Math. Prog. B*, 103:335–356, 2005.

[22] K. Oettershagen. *Ein Superlinear Konvergenter Algorithmus zur Lösung Semi-Infiniter Optimierungsprobleme.* PhD thesis, Bonn University, 1982.

[23] J. Rohn. Positive definiteness and stability of interval matrices. *SIAM J. Matrix Anal. Appl.*, 15(1):175–184, January 1994.

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

[25] M. Tawarmalani and N. V. Sahinidis. *Convexification and Global Optimization in Continuous and Mixed-Integer Nonlinear Programming: Theory, Algorithms, Software, and Applications.* Kluwer, Dordrecht, Netherlands, 2002.

[26] A. Wächter. Homepage of IPOPT, 2002. `http://www.coin-or.org/Ipopt/index.html`.

[27] J. L. Zhou and A. L. Titts. An SQP algorithm for finely discretized continuous minimax problems and other minimax problems with many objective functions. *SIAM J. Optim.*, 6(2):461–487, 1996.