BIT Numer Math (2010) 50: 885–917
DOI 10.1007/s10543-010-0280-6

BIT

Nonsmooth exclusion test for finding all solutions of nonlinear equations

M.D. Stuber · V. Kumar · P.I. Barton

Received: 3 November 2009 / Accepted: 18 August 2010 / Published online: 16 September 2010

Abstract A new approach is proposed for finding all real solutions of systems of nonlinear equations with bound constraints. The zero finding problem is converted to a global optimization problem whose global minima with zero objective value, if any, correspond to all solutions of the original problem. A branch-and-bound algorithm is used with McCormick's nonsmooth convex relaxations to generate lower bounds. An inclusion relation between the solution set of the relaxed problem and that of the original nonconvex problem is established which motivates a method to generate automatically, starting points for a local Newton-type method. A damped-Newton method with *natural level functions* employing the *restrictive monotonicity test* is employed to find solutions robustly and rapidly. Due to the special structure of the objective function, the solution of the convex lower bounding problem yields a nonsmooth root exclusion test which is found to perform better than earlier interval-analysis based exclusion tests. Both the componentwise Krawczyk operator and interval-Newton operator with Gauss-Seidel based root inclusion and exclusion tests are also embedded in the proposed algorithm to refine the variable bounds for efficient fathoming of the search space. The performance of the algorithm on a variety of test problems from the literature is presented, and for most of them, the first solution is found at the first iteration of the algorithm due to the good starting point generation.

Communicated by Hans Petter Langtangen.

M.D. Stuber · P.I. Barton (✉)
Dept. of Chemical Engineering, Massachusetts Institute of Technology, Cambridge, MA, 02139, USA
e-mail: pib@mit.edu

V. Kumar
Computation for Design and Optimization, Massachusetts Institute of Technology, Cambridge, MA, 02139, USA

Keywords Global optimization · Interval analysis · Convex relaxation · Branch-and-bound

Mathematics Subject Classification (2000) 65G40 · 90C57 · 34A34

1 Introduction

One of the most challenging problems arising in many science and engineering applications is the reliable solution of systems of nonlinear equations. This is expressed mathematically as finding $\mathbf{x} \in \mathbb{R}^n$ such that

$$\mathbf{f}(\mathbf{x}) = \mathbf{0} \tag{1}$$

where $\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^n$ is the function describing a certain model. Also, quite often the model under consideration is only valid on a subset of $\mathbb{R}^n$, usually in an n-dimensional box formed by physical bounds on the variables $\mathbf{x}$. Such a box $\mathbb{X}$ is described as the connected compact set

$$\mathbb{X} = \{\mathbf{x} \in \mathbb{R}^n : \mathbf{x}^L \leq \mathbf{x} \leq \mathbf{x}^U\}$$

where $\mathbf{x}^L \in \mathbb{R}^n$ and $\mathbf{x}^U \in \mathbb{R}^n$ are, respectively, the lower and upper bounds on $\mathbf{x}$. The box $\mathbb{X}$ is an *interval vector*, and therefore is an element of the set of all interval subsets of $\mathbb{R}^n$ denoted $\mathbb{IR}^n$. Thus, $\mathbb{X} \subset \mathbb{R}^n$ can equivalently be expressed as the vector $\mathbb{X} \in \mathbb{IR}^n$ whose i^{th} element is X_i or $\mathbb{X}(i)$ with x_i^L and x_i^U as lower and upper bounds, respectively. The algorithm presented in this paper is a *complete* method, in that it finds all isolated real solutions of (1) in a given box $\mathbb{X} \in \mathbb{IR}^n$ with mathematical certainty, as defined in [26].

In practice, *Newton* and *quasi-Newton type methods* are widely applied to (1) to achieve superlinear convergence (see, e.g. [27]), but only locally. Local convergence means that, corresponding to each isolated solution $\mathbf{x}^*$ of (1), there exists an $\epsilon > 0$ defining a neighborhood $N_\epsilon(\mathbf{x}^*)$ of $\mathbf{x}^*$ such that all starting points located in $N_\epsilon(\mathbf{x}^*)$ will generate a sequence of iterates converging to the solution $\mathbf{x}^*$. Attempts have been made to enlarge the neighborhood of convergence, known as "global convergence", by use of step-length control strategies known as *line searches*. This requires approximately minimizing a suitably chosen one-dimensional *merit function*. A common choice for merit function is the squared Euclidean norm of $\mathbf{f}(\mathbf{x})$ which leads to the *damped-Newton method*. However, even for mildly ill-conditioned problems, the damped-Newton method produces extremely small stepsizes leading to very slow convergence. As pointed out by Deuflhard [3], this happens because, for ill-conditioned problems, the Newton direction and the steepest-descent direction of this merit function are almost orthogonal. This observation motivates the choice of the *natural level function* as a merit function [3] which gives a steepest-descent direction parallel to the Newton direction. However, the natural level function changes at each iteration and so traditional descent arguments can no longer be used to prove global convergence. This led Bock *et al.* [1] to propose the *restrictive monotonicity*

test (RMT) which is essentially an alternative stepsize selection strategy to exact or approximate line searches using natural level functions.

Although these efforts have significantly enlarged the region of convergence, finding a close enough "starting point" still remains a nontrivial task, as practical problems are often large, highly nonlinear and ill-conditioned and therefore exhibit a very small neighborhood of convergence. In practice, a large amount of time can be expended trying to find a suitable starting point using a variety of ad hoc strategies. An important contribution of the algorithm proposed here is the development of a reliable technique to generate automatically starting points which are in a sense "reasonably close" to the solutions sought.

Interval Newton-type methods apply a modified form of real-valued Newton-type methods to interval-valued variables. Coupled with a generalized bisection strategy [7], initial intervals are refined and converge to boxes that enclose all isolated (real) solutions of (1). A detailed discussion of interval-based methods for solving systems of equations can be found in the monograph by Neumaier [25]. Such methods usually have operators under different names (e.g., interval-Newton, Krawczyk) which are interval-valued functions of interval- and real-valued variables. While the interval iteration schemes for solving systems of nonlinear equations have been used to obtain tight enclosures of the solutions, what is most significant is the ability to provide an existence and uniqueness test known as a *root inclusion test*. Furthermore, there are also many *root exclusion tests* with which one can ascertain that no solution for the system of equations exists in the given box. The interval-Newton operator can compute tight enclosures of the solution and its calculation simply requires the solution of an interval linear system. A Gauss-Seidel implementation offers an efficient way to calculate the solution, however, it requires division by an interval matrix component that may contain zero. Although extended interval arithmetic may then be employed, there are also negative implications with doing so. On the other hand, the less-tight Krawczyk operator avoids such divisions and therefore may be preferred over the interval-Newton operator for some problems. Almost always, preconditioning matrices are desired or required in the computation of the interval operators, and the strength of the associated exclusion and inclusion tests depends on the quality of such preconditioning.

In the proposed algorithm, once a solution is found, the Krawczyk or interval-Newton [25] root inclusion test is applied to fathom a region of the search space which is known to contain this unique solution. Also, since the solution has been found prior to the application of the inclusion test, excellent preconditioning is achieved using the inverse of the Jacobian matrix evaluated at the solution. The interval-Newton and Krawczyk's root exclusion tests are also embedded in the proposed algorithm, which help in refining the variable bounds and in fathoming regions not containing a solution. Interval Newton-type methods have been successfully applied by Schnepper and Stadtherr [28] to find all solutions of nonlinear equations with bound constraints. Their approach is an improvement of the interval-Newton/generalized bisection strategy of Kearfott [10] with suitable modifications in the bisection strategy and preconditioner computation and storage. It finds boxes which contain the solutions with mathematical certainty.

A global optimization reformulation of the root finding problem has been used successfully by Maranas and Floudas [18] to find all solutions of systems of nonlin-

ear equations. In the proposed algorithm, an approach similar to [18] is used in that the original problem is converted to a global optimization problem with the objective function taken as a suitable norm of $\mathbf{f}(\mathbf{x})$. Most general, deterministic strategies applied to solve nonconvex programs involve solving a corresponding convex program in which the objective function is a *convex relaxation* of the original nonconvex function. The optimal value of this convex program is always a lower bound on the optimal value of the original nonconvex program on the chosen set. McCormick [19] has proposed an automatic, recursive procedure for constructing the convex relaxation of any elementary function that can be implemented as a computer program. In most cases, convex relaxations constructed using McCormick's procedure are found to be tighter than those obtained by other alternative approaches (such as αBB [18], interval extension [25]) and hence McCormick's procedure has been used in the proposed algorithm.

As will be shown later, due to the special structure of the objective function, the McCormick relaxation has certain properties leading to an inclusion relation between the solution set of the convex relaxation and the set of solutions of (1). This inclusion relation is exploited to generate starting points automatically for local Newton-type iterations. Not only this, the presented technique also yields a *root exclusion test* to verify that no admissible solution of (1) exists in the set under consideration. This feature proves extremely useful in its own right in accelerating the convergence of the algorithm and in general can also be used to debug poorly formulated models and/or simulation problems. Moreover, this so-called nonsmooth root exclusion test is found to outperform both the Krawczyk exclusion test and the interval-Newton with Gauss-Seidel exclusion tests on a variety of test problems from the literature.

In general, the convex relaxations generated using McCormick's method are nonsmooth (i.e., not differentiable on their domains) and hence standard convex optimization algorithms which assume differentiability cannot be applied. However, there is a class of nonsmooth optimization methods available in the literature, such as *bundle methods* [14, 17] and the *variable metric method* [15], which can be applied to solve the resulting nonsmooth convex program, provided the objective function value and a subgradient can be evaluated at all desired points.

A number of algorithms reported in the literature, including Wilhem and Swaney [29], whose algorithm finds a single solution, and Kearfott [9], whose algorithm finds all solutions utilizing interval-Newton existence tests within a generalized bisection algorithm, have embedded a Newton-type method within a branch-and-bound framework for solving systems of nonlinear equations. The proposed algorithm does this by integrating RMT damped-Newton iterations with a convex lower bounding strategy. This is motivated by the automatic generation of a reasonably good starting point as a result of solving the convex lower bounding problem and hence often leads to fast convergence to the solutions sought using a real-valued Newton-type method. The algorithm applies a RMT-based subroutine NWTSLV, after a suitable starting point is computed, and in most of the test problems NWTSLV is found to converge to a solution from the generated starting point. The efficacy of this feature is evidenced from the fact that the first solution to most of the test problems is found at the very first iteration of the algorithm. This approach is particularly helpful when only one admissible solution to (1) is required. For example, when attempting to solve suffi-

ciently large and/or complex problems, finding all solutions is not always necessary or even tractable.

Homotopy-continuation [5] and global terrain [13] are two other global root finding methods of a different class. Homotopy-continuation follows a smooth solution path from an arbitrary initial guess to a solution of (1). Whereas global terrain intelligently surveys a function surface relying on the fact that solutions and singular points are connected by smooth paths. Both methods benefit from having the ability to calculate real and complex solutions to (1). However, homotopy-continuation falls short of locating all solutions when there exist multiple (disconnected) solution paths within the feasible region. Similarly, due to the heuristic termination criterion that global terrain relies on, it may too fail to find all solutions prior to termination. Although the proposed method cannot locate complex solutions, the trade-off is that it is a complete method, and therefore offers the ability to guarantee that all isolated real solutions have been located (to within ϵ tolerance) within the box $\mathbb{X}$ upon termination.

2 Problem formulation

The root finding problem described by (1) can be reformulated as the following global optimization problem over the admissible domain $\mathbb{X} \in \mathbb{IR}^n$:

$$\min_{\mathbf{x}\in\mathbb{X}} \|\mathbf{f}(\mathbf{x})\|$$

where $\|.\|$ can be any vector norm. Numerical considerations dictate that the 1-norm is the best choice in the presented algorithmic framework, as will be discussed in Sect. 3.2. This results in the following optimization problem:

$$\min_{\mathbf{x}\in\mathbb{X}} \sum_{i=1}^{n} |f_i(\mathbf{x})|. \tag{2}$$

If the optimal value of the above optimization problem is zero, any corresponding $\mathbf{x}$ will be a solution of (1). In fact, the global optimal solutions with zero optimal value of (2) are equivalent to the admissible solution set of (1).

However, in general (2) is a nonconvex program and a local solver cannot guarantee convergence to a global optimal solution. Hence, for such problems, one strategy is to construct a *convex relaxation* of the objective function and solve the resulting convex program to generate a lower bound on the global optimal value.

Definition 1 (Convex/concave relaxation) Let $u, o, \phi : C \subset \mathbb{R}^n \to \mathbb{R}$ with C nonempty and convex. Suppose u is convex and o is concave on C. Then u is a *convex relaxation* of ϕ on C if

$$u(\mathbf{x}) \leq \phi(\mathbf{x}), \quad \forall \mathbf{x} \in C,$$

and o is a *concave relaxation* of ϕ on C if

$$o(\mathbf{x}) \geq \phi(\mathbf{x}), \quad \forall \mathbf{x} \in C.$$

Convex and concave relaxations are quite frequently used in global optimization algorithms and given a function, a variety of techniques are known for constructing these relaxations. In the proposed algorithm McCormick's procedure [19] is used for the construction of convex relaxations as discussed below.

2.1 McCormick's composition theorem

McCormick [19] has proposed a method for constructing convex and concave relaxations of a function $F[f(\mathbf{x})]$ defined by the composition of a multivariate function f with a univariate function F. The following theorem, known as McCormick's composition theorem, enables the construction of convex and concave relaxations of the composition.

Theorem 1 (McCormick's composition theorem) *Let $C \subset \mathbb{R}^n$ be a nonempty convex set. Consider the composite function $F \circ f$ where $f : C \to \mathbb{R}$ is continuous, and let $f(C) \subset [a, b]$. Suppose that a convex function c^u and a concave function c^o satisfying*

$$c^u(\mathbf{x}) \leq f(\mathbf{x}) \leq c^o(\mathbf{x}), \quad \forall \mathbf{x} \in C$$

are known. Let $e : [a, b] \to \mathbb{R}$ be a convex relaxation of F on $[a, b]$ and let $E : [a, b] \to \mathbb{R}$ be a concave relaxation of F on $[a, b]$. Let z_{min} be point at which e attains its infimum on $[a, b]$ and let z_{max} be point at which E attains its supremum on $[a, b]$. If the above conditions hold, then

$$u(\mathbf{x}) = e[\text{mid}\{c^u(\mathbf{x}), c^o(\mathbf{x}), z_{min}\}]$$

is a convex relaxation of $F \circ f$ on C and,

$$o(\mathbf{x}) = E[\text{mid}\{c^u(\mathbf{x}), c^o(\mathbf{x}), z_{max}\}]$$

is a concave relaxation of $F \circ f$ on C, where the mid *function selects the middle value of the three scalars.*

The theorem requires prior knowledge of a and b such that $f(C) \subset [a, b]$. This can be done by taking the *natural interval extension* [25] of f on a box $\mathbb{X} \supset C$ and using f^L and f^U for a and b, respectively. Hence, the relaxations become dependent on the strength of the interval extensions and so a weak interval extension may result in weak relaxations.

2.2 Convex relaxations of factorable functions

Factorable functions are similar to the class of functions for which natural interval extensions can be calculated.

Definition 2 (Factorable function) Let $f : C \subset \mathbb{R}^n \to \mathbb{R}$ with C nonempty and convex. Then f is a *factorable function* if it can be expressed in terms of a finite sequence of factors $v_1, \ldots, v_N$ such that, given $\mathbf{x} \in C$, $v_k = x_k$ for $i = 1, \ldots, n$, and for $i = n + 1, \ldots, N$ as

1. $v_i = v_j + v_k,\ \ j,k < i$ or,
2. $v_i = v_j v_k,\ \ j,k < i$ or,
3. $v_i = F(v_j),\ \ j < i$ where $F(\cdot)$ is a univariate intrinsic function,

and $f(\mathbf{x}) = v_N(\mathbf{x})$.

Subtraction and division operations can be handled by introducing univariate negative and reciprocal intrinsic functions. Most of the functions that are implementable as a computer program are factorable in the above sense. There are usually several different representations of a given function as factors, and different representations may yield different convex and concave relaxations. For example, x^2 can be treated as a univariate intrinsic function or as a binary multiplication to yield two different relaxations.

If the interval extensions and convex and concave relaxations of two functions are known on a given interval, the corresponding relaxations for their sum, difference and product can be easily computed. For example, to compute a relaxation of the binary product $f_1 f_2$ on C having c_i^u and $c_i^o,\ i = 1, 2$ as their convex and concave relaxations, respectively, it is required to find numbers $f_1^L, f_1^U, f_2^L, f_2^U$ (usually obtained from the natural interval extensions) such that

$$C \subset \{\mathbf{x} : f_1^L \le f_1(\mathbf{x}) \le f_1^U, f_2^L \le f_2(\mathbf{x}) \le f_2^U\}.$$

From the convex envelope of a bilinear function it is known that

$$f_1(\mathbf{x}) f_2(\mathbf{x}) \ge \max\{f_2^L f_1(\mathbf{x}) + f_1^L f_2(\mathbf{x}) - f_1^L f_2^L, f_2^U f_1(\mathbf{x}) + f_1^U f_2(\mathbf{x}) - f_1^U f_2^U\},$$
$$\forall \mathbf{x} \in C. \tag{3}$$

Now defining,

$$\alpha_1(\mathbf{x}) \equiv \min\{f_2^L c_1^u(\mathbf{x}), f_2^L c_1^o(\mathbf{x})\},$$
$$\alpha_2(\mathbf{x}) \equiv \min\{f_1^L c_2^u(\mathbf{x}), f_1^L c_2^o(\mathbf{x})\},$$
$$\beta_1(\mathbf{x}) \equiv \min\{f_2^U c_1^u(\mathbf{x}), f_2^U c_1^o(\mathbf{x})\},$$
$$\beta_2(\mathbf{x}) \equiv \min\{f_1^U c_2^u(\mathbf{x}), f_1^U c_2^o(\mathbf{x})\},$$

it can be verified that the functions $\alpha_1, \alpha_2, \beta_1$ and β_2 are convex on C. Also, it follows from (3) that

$$f_1(\mathbf{x}) f_2(\mathbf{x}) \ge \max\{\alpha_1(\mathbf{x}) + \alpha_2(\mathbf{x}) - f_1^L f_2^L, \beta_1(\mathbf{x}) + \beta_2(\mathbf{x}) - f_1^U f_2^U\}, \quad \forall \mathbf{x} \in C. \tag{4}$$

Each argument in the max function on the RHS of (4) is convex and the maximum of two convex functions is convex, making it a valid convex relaxation of $f_1 f_2$ on C. A parallel argument can be developed for computing the concave relaxation of the binary product form.

In order to construct convex and concave relaxations of a factorable function f on C, given an interval vector (box) $\mathbb{X} \supset C$, set the first $i = 1, 2, \ldots, n$ convex and

concave relaxations as:

$$c_i^u = x_i,$$
$$c_i^o = x_i$$

and the interval extensions $i = 1, 2, \ldots, n$ as

$$V_i = X_i,$$

where again, X_i are the elements of the interval vector $\mathbb{X}$. Assuming convex and concave relaxations are known for all univariate intrinsic functions from which f is composed, each factor $i = n+1, \ldots, N$ can be augmented with expressions defining its concave and convex relaxations, using the rules for univariate intrinsic functions, binary addition and binary multiplication. Each factor $i = n+1, \ldots, N$ can also be augmented with the expression for its natural interval extension in order to propagate the bounds needed in the expressions for the relaxations. This in fact defines a sequence of statements that can be executed by a computer program in order to evaluate simultaneously:

1. f at $\mathbf{x}$,
2. the required convex and concave relaxations on $\mathbb{X}$ at $\mathbf{x}$, and
3. the natural interval extension of f on the box $\mathbb{X}$.

Hence, the relaxations can be easily implemented as a computer program using the operator overloading features of modern programming languages. The running time required will be a small fixed multiple of the running time required to evaluate the original factorable representation of the function.

It is evident that, due to the frequent occurrence of min, max and mid terms in the expressions for evaluating McCormick's relaxations, these relaxations are usually nonsmooth and so convex optimization methods assuming differentiability cannot be applied to solve the resulting convex program. However, nonsmooth optimization techniques employing subgradients of the objective function can be applied. Recently, a method to compute subgradients of McCormick's convex relaxations has also been developed [20] which works in a very similar manner to the way in which automatic differentiation computes the gradient of a smooth function [4]. This enables the resulting convex program to be solved using nonsmooth optimization methods such as *bundle methods* [14, 17] and the *variable metric method* [15]. The Fortran codes [16] implementing the variable metric method (PVARS) and the proximal bundle method (PBUNS) work quite well on McCormick's nonsmooth functions and have been used in the implementation of the presented algorithm. McCormick's convex relaxation of the objective function in (2) is computed to define the lower bounding convex program, a solution of which gives a root exclusion test as well as a reliable technique to automatically generate a starting point for a real-valued Newton-type iteration.

3 Theoretical development

Consider the nonconvex minimization problem (2). By constructing McCormick's convex underestimator $u : \mathbb{X} \to \mathbb{R}$ of the objective function over $\mathbb{X}$, the following

convex program

$$\min_{\mathbf{x}\in\mathbb{X}} u(\mathbf{x}) \tag{5}$$

can be formulated and solved to bound the optimal value of (2). Since this convex program is obtained using McCormick's convex underestimators, it will be referred to as McCormick's convex program subsequently in this paper.

3.1 Nonsmooth root exclusion test

Theorem 2 (Nonnegativity) *Let* $\mathbf{f}:\mathbb{X}\to\mathbb{R}^n$ *be any factorable function defined on an* n*-dimensional box* $\mathbb{X}$*. Then, the McCormick's convex relaxation* $u:\mathbb{X}\to\mathbb{R}$ *underestimating the vector* 1*-norm* $\|\mathbf{f}(\cdot)\|_1$ *of* $\mathbf{f}$ *is nonnegative on* $\mathbb{X}$*.*

Proof Let c_i^u and c_i^o, respectively, denote convex and concave relaxations of function f_i on $\mathbb{X}$ for $i=1,2,\ldots,n$. Thus,

$$c_i^u(\mathbf{x}) \le f_i(\mathbf{x}) \le c_i^o(\mathbf{x}), \quad \forall \mathbf{x}\in\mathbb{X},\ i=1,2,\ldots,n.$$

Also suppose that the numbers f_i^L and f_i^U are known (e.g., from natural interval extensions) such that

$$f_i^L \le f_i(\mathbf{x}) \le f_i^U, \quad \forall \mathbf{x}\in\mathbb{X},\ i=1,2,\ldots,n.$$

To construct the convex relaxation u_i of $|f_i(\cdot)|$ over $\mathbb{X}$ we observe that the univariate outer function is $F(z)=|z|$ which being convex is its own convex relaxation and attains its infimum at $z_{i,min}=\text{mid}\{f_i^L, f_i^U, 0\}$ over $[f_i^L, f_i^U]$ for $i=1,2,\ldots,n$. Hence, using McCormick's composition theorem the convex underestimator of $|f_i(\cdot)|$ over $\mathbb{X}$ will be given by

$$u_i(\mathbf{x}) = |\text{mid}\{c_i^u(\mathbf{x}), c_i^o(\mathbf{x}), z_{i,min}\}| \ge 0, \quad i=1,2,\ldots,n. \tag{6}$$

Now, the convex relaxation u for the 1-norm of $\mathbf{f}$ can be obtained by adding the individual convex underestimators u_i, as summation of convex functions preserves convexity. Hence,

$$u(\mathbf{x}) = \sum_{i=1}^{n} u_i(\mathbf{x}) \le \sum_{i=1}^{n} |f_i(\mathbf{x})|, \quad \forall \mathbf{x}\in\mathbb{X}.$$

Using (6), all the terms involved in the left summation above are nonnegative thereby making u nonnegative on $\mathbb{X}$. □

The theorem above is proved only for the 1-norm of $\mathbf{f}$ but can be easily generalized to any norm. It implies that the optimal value of the convex program (5) is nonnegative. Thus, if (5) is solved and its optimal value is found to be positive, then based on the underestimating property of the convex relaxation, it is concluded that no solution to (1) exists in $\mathbb{X}$. This so-called *root exclusion test* provides a rigorous method to verify that no admissible solution to $\mathbf{f}(\mathbf{x})=\mathbf{0}$ exists and proves extremely

useful to debug poorly formulated models and/or simulation problems. In the proposed branch-and-bound algorithm for solving systems of nonlinear equations this test is used to fathom a large part of the search space.

3.2 Automatic starting point generation

If the optimal value of (5) is found to be zero, it does not necessarily imply that an admissible solution of (1) exists. Nevertheless, the solution point found can be used as an automatically generated starting point for a Newton-type iteration to find a solution to (1), if it exists. The following theorem motivates this choice of starting point.

Theorem 3 (Solution set inclusion) *Let* $\mathbf{f}:\mathbb{X}\to\mathbb{R}^n$ *be any factorable function defined on an n-dimensional box* $\mathbb{X}$ *and let the sets* S *and* U *be defined as*

$$S=\{\mathbf{x}\in\mathbb{X}:\mathbf{f}(\mathbf{x})=\mathbf{0}\},$$
$$U=\{\mathbf{x}\in\mathbb{X}:u(\mathbf{x})=0\}$$

where u *is the McCormick convex relaxation of the* 1*-norm* $\|\mathbf{f}(\cdot)\|_1$ *of* $\mathbf{f}$ *on* $\mathbb{X}$*. Then,* $S\subset\mathrm{conv}(S)\subset U\subset\mathbb{X}$*, where* $\mathrm{conv}(S)$ *denotes the convex hull of the set* S*.*

Proof Let $\hat{\mathbf{x}}\in\mathrm{conv}(S)$. Using Caratheodory's theorem, there exist scalars $\lambda_j\geq 0$ and $\mathbf{x}_j\in S$ for $j=1,2,\ldots,n+1$, such that

$$\hat{\mathbf{x}}=\sum_{j=1}^{n+1}\lambda_j\mathbf{x}_j\quad\text{and}\quad\sum_{j=1}^{n+1}\lambda_j=1.$$

Again, any $\mathbf{x}^*\in S$ will also be a zero optimal solution of the optimization problem

$$\min_{\mathbf{x}\in\mathbb{X}}\|\mathbf{f}(\mathbf{x})\|_1=\min_{\mathbf{x}\in\mathbb{X}}\sum_{i=1}^{n}|f_i(\mathbf{x})|.$$

Using the nonnegativity and underestimating property of u, it follows that

$$0\leq u(\mathbf{x}^*)\leq\sum_{i=1}^{n}|f_i(\mathbf{x}^*)|=0,\quad\forall\mathbf{x}^*\in S$$
$$\Rightarrow\quad u(\mathbf{x}^*)=0,\quad\forall\mathbf{x}^*\in S.$$

Again nonnegativity and convexity of u gives

$$0\leq u(\hat{\mathbf{x}})=u\left(\sum_{j=1}^{n+1}\lambda_j\mathbf{x}_j\right)\leq\sum_{j=1}^{n+1}\lambda_j u(\mathbf{x}_j)=0$$
$$\Rightarrow\quad u(\hat{\mathbf{x}})=0\quad\Rightarrow\quad\hat{\mathbf{x}}\in U\quad\Rightarrow\quad\mathrm{conv}(S)\subset U.\qquad\square$$

As per the above theorem, the automatically generated starting point will lie in the set U which contains the convex hull of the desired solution set S. If the number of admissible solutions to (1) is small as compared to the dimension n (a reasonable expectation for engineering problems), then conv(S) will be a smaller set relative to $\mathbb{X}$. Also, if U is not much larger than conv(S), any of the points in U is likely to be close to an admissible solution of (1). In fact, the difference of these two sets bear a close relation with the tightness of the convex relaxation $u(\mathbf{x})$.

Continuing with the notations used in the proof of Theorem 2, let c_i^u and c_i^o be convex and concave relaxations of $f_i(\mathbf{x})$ on $\mathbb{X}$, respectively, and u_i denotes the McCormick convex relaxation of $|f_i(\cdot)|$ for $i = 1, 2, \ldots, n$. The convex relaxation u of $\|\mathbf{f}(\cdot)\|_1$ will be

$$u(\mathbf{x}) = \sum_{i=1}^{n} u_i(\mathbf{x}). \tag{7}$$

By definition $\mathbf{x}^* \in U \Rightarrow u(\mathbf{x}^*) = 0$. Nonnegativity of u_i together with (7) imply that for $i = 1, 2, \ldots, n$

$$u_i(\mathbf{x}^*) = 0 \Rightarrow |\text{mid}\{c_i^u(\mathbf{x}^*), c_i^o(\mathbf{x}^*), z_{i,min}\}| = 0. \tag{8}$$

Now, assuming that at least one admissible solution of $\mathbf{f}(\mathbf{x}) = \mathbf{0}$ exists, the following necessary condition

$$f_i^L \leq 0 \leq f_i^U, \quad \forall i = 1, 2, \ldots, n,$$

will always hold true over $\mathbb{X}$. This implies,

$$z_{i,min} = \text{mid}\{f_i^L, f_i^U, 0\} = 0, \quad \forall i = 1, 2, \ldots, n.$$

Also, by construction $c_i^u(\mathbf{x}^*) \leq c_i^o(\mathbf{x}^*)$ and in light of the above assumption only one of the following holds true:

$$c_i^u(\mathbf{x}^*) \leq 0 \leq c_i^o(\mathbf{x}^*), \tag{9}$$

$$0 < c_i^u(\mathbf{x}^*) \leq c_i^o(\mathbf{x}^*), \tag{10}$$

$$c_i^u(\mathbf{x}^*) \leq c_i^o(\mathbf{x}^*) < 0. \tag{11}$$

The last two inequality relations (10) and (11) imply $u_i(\mathbf{x}^*) \neq 0$ leaving (9) which clearly asserts $u_i(\mathbf{x}^*) = 0$ as per (8). Hence, the set U can be alternatively described as

$$U = \{\mathbf{x}^* \in \mathbb{X} : c_i^u(\mathbf{x}^*) \leq 0, -c_i^o(\mathbf{x}^*) \leq 0, \ i = 1, 2, \ldots, n\}. \tag{12}$$

This characterization of U further motivates the choice of starting point to be a point lying inside it. The concave $c_i^o(\mathbf{x}^*)$ and convex $c_i^u(\mathbf{x}^*)$ relaxations of the functions $f_i(\mathbf{x})$ at any point in $\mathbf{x}^* \in U$ are opposite in sign for $i = 1, 2, \ldots, n$, making it likely for $\mathbf{x}^*$ to be close to a point in the solution set S. Moreover, U being the solution set of a convex program, it is expected to be a convex set, as confirmed by the above characterization (12), where it is represented explicitly by inequality constraints involving convex functions.

Furthermore, even if no admissible solution of (1) exists, there is the possibility that in the computed natural interval extension of $\mathbf{f}$ over $\mathbb{X}$, $f_i^L \leq 0$ and $f_i^U \geq 0$, $\forall i = 1, 2, \ldots, n$. Hence, the natural interval extension will not be able to detect the nonexistence of a root. Moreover, in this case $z_{i,min} = 0$, $\forall i = 1, 2, \ldots, n$. Assume that for any $1 \leq i \leq n$ either $c_i^u(\mathbf{x}) > 0$ or $c_i^o(\mathbf{x}) < 0$, $\forall \mathbf{x} \in \mathbb{X}$. Thus, $u(\mathbf{x}) > 0$, $\forall \mathbf{x} \in \mathbb{X}$ and the nonsmooth exclusion test can detect nonexistence of a root.

In the proposed branch-and-bound algorithm, convex relaxations and subgradients are calculated automatically using the library libMC [2] and the nonsmooth solver PVARS [16] is used to solve McCormick's convex program (5). Assuming that S is nonempty over the current box $\mathbb{X}$, the solver is supposed to find a point in the set U with an optimal value of zero. It can be easily deduced that, in theory, the set U is invariant for any choice of 1, 2 or ∞ norms of $\mathbf{f}$. However, numerical solvers rely on pre-specified tolerances and thresholds for termination and so, numerically there is a larger set $\hat{U}$ enclosing the real set U within which the nonsmooth solver is likely to converge. The convex underestimators are expected to be flat around the convex hull of the solution set S and will be further flattened on squaring thereby making the enclosing set $\hat{U}$ larger. Hence, the most obvious choice of squared Euclidean norm of $\mathbf{f}(\mathbf{x})$ as the objective function is not suitable on account of the above numerical consideration. Out of the other two choices, the 1-norm is preferred over the infinity-norm to make the exclusion test more effective. In the infinity-norm only one of the functions out of $f_i(\mathbf{x})$, $i = 1, 2, \ldots, n$ will be contributing to the optimal value of the underestimating convex program, making it a poor choice compared to 1-norm where all the n functions will make their contribution.

3.3 Interval Newton-type root inclusion & exclusion tests

Interval methods for bounding the solutions of systems of equations have been the focus of much research [21–23, 25, 28], primarily because of their intrinsic ability to guarantee whether or not a solution exists within the current interval of interest, or *box*. The so-called *inclusion tests* check the existence and uniqueness of a solution in a box, whereas the *exclusion tests* tell us when there are no solutions in the box. Two tests, based on the interval-Newton and Krawczyk operators, are considered in this paper. There are advantages and disadvantages of each. The Krawczyk operator is defined in the following.

Definition 3 (Krawczyk) For an interval $\mathbb{X} \in \mathbb{IR}^n$, a point $\mathbf{x} \in \mathbb{X}$ and a function $\mathbf{f} : D \to \mathbb{R}^n$ continuously differentiable on the open set $D \subset \mathbb{R}^n$ with $\mathbb{X} \subset D$, the Krawczyk operator is defined as

$$\mathbb{K}(\mathbf{x}, \mathbf{f}, \mathbb{X}) \equiv \mathbf{x} - \mathbf{Y}\mathbf{f}(\mathbf{x}) + (\mathbf{I} - \mathbf{Y}\mathbb{J}(\mathbf{f}, \mathbb{X}))(\mathbb{X} - \mathbf{x}), \tag{13}$$

where $\mathbf{Y} \in \mathbb{R}^{n \times n}$ is a linear isomorphism used for preconditioning, $\mathbf{I} \in \mathbb{R}^{n \times n}$ is the $n \times n$ identity matrix, and $\mathbb{J}(\mathbf{f}, \mathbb{X}) \in \mathbb{IR}^{n \times n}$, is an interval extension of the Jacobian matrix of $\mathbf{f}$ on $\mathbb{X}$.

For better convergence and enclosure properties, a componentwise implementation of $\mathbb{K}$ is recommended [6]. That is, where each element of $\mathbb{K}$ is calculated se-

quentially making use of the previously calculated elements. The componentwise Krawczyk operator is defined as:

Definition 4 (Componentwise Krawczyk)

$$\begin{aligned} K_i^k &:= x_i^k - b_i + \sum_{j=1}^{i-1} A_{ij}(X_j^{k+1} - x_j^k) + \sum_{j=i}^{n} A_{ij}(X_j^k - x_j^k), \\ X_i^{k+1} &:= K_i^k \cap X_i^k, \end{aligned} \tag{14}$$

with k being the current iteration or calculation, i and j denoting particular elements of the vector or matrix, $\mathbb{A} \in \mathbb{IR}^{n\times n}$ as $\mathbb{A} = \mathbf{I} - \mathbf{Y}\mathbb{J}(\mathbf{f}, \mathbb{X})$, and $\mathbf{b} \in \mathbb{R}^n$ as $\mathbf{b} = \mathbf{Yf}(\mathbf{x})$.

The other powerful operator considered in this paper is the interval-Newton operator defined in the following.

Definition 5 (Interval-Newton) For an interval $\mathbb{X} \in \mathbb{IR}^n$, a point $\mathbf{x} \in \mathbb{X}$ and a function $\mathbf{f} : D \to \mathbb{R}^n$ continuously differentiable on an open set $D \subset \mathbb{R}^n$ with $\mathbb{X} \subset D$, the interval Newton operator is defined as

$$\mathbb{N}(\mathbf{x}, \mathbf{f}, \mathbb{X}) \equiv \mathbf{x} - \mathbb{J}(\mathbf{f}, \mathbb{X})^{-1}\mathbf{f}(\mathbf{x}), \tag{15}$$

with $\mathbb{J}(\mathbf{f}, \mathbb{X})$ being an interval extension of the Jacobian matrix of $\mathbf{f}$ on $\mathbb{X}$.

It is assumed that $\mathbb{J}$ does not contain any singular matrices. Similar to the Krawczyk operator, a componentwise sequential calculation of $\mathbb{N}$ offers better convergence and enclosure properties. The componentwise implementation is known as the generalized interval *Gauss-Seidel* (G-S) method. It is a result of arranging (15) into a linear system $\mathbf{A}\Delta\mathbf{x} = -\mathbf{b}$ as

$$\mathbb{J}(\mathbf{f}, \mathbb{X})(\mathbb{N}(\mathbf{x}, \mathbf{f}, \mathbb{X}) - \mathbf{x}) = -\mathbf{f}(\mathbf{x})$$

and solving for $\mathbb{N}$ componentwise. In [25], it is proven that $\mathbb{N} \subset \mathbb{K}$ when $\mathbb{N}$ is implemented with G-S. The G-S implementation is defined in the following.

Definition 6 (Interval-Newton with Gauss-Seidel)

$$\begin{aligned} N_i^k &:= x_i^k - \left[b_i + \sum_{j=1}^{i-1} A_{ij}(X_j^{k+1} - x_j^{k+1}) + \sum_{j=i+1}^{n} A_{ij}(X_j^k - x_j^k) \right] / A_{ii}, \\ 0 &\notin A_{ii},\ X_i^{k+1} := N_i^k \cap X_i^k, \end{aligned} \tag{16}$$

where $\mathbb{A} \in \mathbb{IR}^{n\times n}$ as $\mathbb{A} = \mathbf{Y}\mathbb{J}(\mathbf{x}, \mathbf{f}, \mathbb{X})$, and $\mathbf{b} \in \mathbb{R}^n$ as $\mathbf{b} = \mathbf{Yf}(\mathbf{x})$.

It should be noted that we have again made use of a preconditioning matrix $\mathbf{Y}$ in the above definition, which will help give better numerical behavior of the algo-

rithm. Also, it should be noted that if $0 \in A_{ii}$, extended interval arithmetic can be employed, however, uniqueness of solutions cannot be guaranteed. It is assumed, for this reason, that $0 \notin A_{ii}$, else the inclusion test simply fails to tell us anything about the uniqueness of solutions on the current interval, and thus must be further refined.

Let $\Phi \in \{\mathbb{N}, \mathbb{K}\}$ be a generic interval Newton-type operator. The root inclusion test states that if $\Phi(\mathbf{x}, \mathbf{f}, \mathbb{X}) \subset \mathbb{X}$, then $\mathbb{X}$ contains a unique solution of the system of equations $\mathbf{f}(\mathbf{x}) = \mathbf{0}$ [21, 22]. Furthermore, all solutions of $\mathbf{f}(\mathbf{x}) = \mathbf{0}$ in $\mathbb{X}$, if any, are contained in the intersection $\Phi(\mathbf{x}, \mathbf{f}, \mathbb{X}) \cap \mathbb{X}$. Immediately following is the exclusion test, which states that if $\Phi(\mathbf{x}, \mathbf{f}, \mathbb{X}) \cap \mathbb{X} = \emptyset$, then no solution exists in $\mathbb{X}$ [21, 22]. A componentwise implementation, as previously discussed, offers one obvious improvement in performance since the exclusion test only requires a single element of Φ to have an empty intersection to be true. Other improvements come from the ability to generate tighter enclosures than the standard naive implementations.

For the inclusion test, an ideal preconditioner $\mathbf{Y}$ is the inverse of the Jacobian matrix evaluated at the solution [8, 9, 25]. However, in the interval Newton-type methods, the solution is not known a priori and hence $\mathbf{Y}$ is usually approximated by taking the inverse of the midpoint of the interval matrix $\mathbb{J}(\mathbf{f}, \mathbb{X})$ obtained from the interval extension of the Jacobian [8, 9, 25]. Other effective choices for $\mathbf{Y}$ are also discussed in [8, 9, 25]. In the proposed B&B algorithm, once a solution is found by a real-valued Newton-type method, the Krawczyk or interval-Newton operator is used only to check the uniqueness of the obtained solution in the present box $\mathbb{X}$. Hence, excellent preconditioning is achieved by using the inverted Jacobian matrix at the solution, making the test quite effective. If the root inclusion test is positive the current box can be fathomed based on the uniqueness result. Moreover, the intersection relation itself helps to fathom a good part of the search space not containing any solution. However, for sufficiently wide intervals, the interval Jacobian may contain singular matrices and so the inclusion test fails to tell us anything about uniqueness as well as fails to fathom part of the search space. For the exclusion test, the inverse of the mid-point of the interval Jacobian matrix $\mathbb{J}(\mathbf{f}, \mathbb{X})$ is also used as the preconditioner $\mathbf{Y}$.

Before proceeding to the next section which describes the steps of the proposed branch-and-bound (B&B) algorithm, the theoretical developments made so far in this paper will be demonstrated with the help of the following example.

Example 1 Global minima of the Himmelblau function:

$$\begin{aligned} f_1(x_1, x_2) &= x_1^2 + x_2 - 11 = 0, \\ f_2(x_1, x_2) &= x_1 + x_2^2 - 7 = 0, \\ (x_1, x_2) &\in \mathbb{X} = [-6, 6]^2. \end{aligned}$$

The Himmelblau function is defined as the squared-Euclidean norm of the above system of equations and hence is nonnegative. Thus, the global minima of the Himmelblau function are equivalent to the roots of the above system of equations. In order to construct the McCormick's convex relaxation of the 1-norm of $\mathbf{f}$ it is observed that the natural interval extensions of both f_1 and f_2 over $\mathbb{X}$ contain 0 and hence the infimum

of the outer absolute value function $|\cdot|$ is attained at 0. The following intermediate steps can also be easily verified:

$$
\begin{aligned}
c_1^u(x_1, x_2) &= x_1^2 + x_2 - 11, \\
c_1^o(x_1, x_2) &= x_2 + 25, \\
u_1(x_1, x_2) &= |\mathrm{mid}\{x_1^2 + x_2 - 11, x_2 + 25, 0\}|, \\
c_2^u(x_1, x_2) &= x_1 + x_2^2 - 7, \\
c_2^o(x_1, x_2) &= x_1 + 29, \\
u_2(x_1, x_2) &= |\mathrm{mid}\{x_1 + x_2^2 - 7, x_1 + 29, 0\}|.
\end{aligned}
$$

Hence the McCormick's convex relaxation u of $\|\mathbf{f}(\cdot)\|_1$ on $\mathbb{X}$ is given by

$$
u(x_1, x_2) = |\mathrm{mid}\{x_1^2 + x_2 - 11, x_2 + 25, 0\}| + |\mathrm{mid}\{x_1 + x_2^2 - 7, x_1 + 29, 0\}|.
$$

Figure 1 shows the plot of $\|\mathbf{f}(\cdot)\|_1$ and its McCormick's convex relaxation u. There are four points at which $\|\mathbf{f}(\cdot)\|_1$ touches the $z = 0$ plane marking the four roots of the system of equations in Example 1. The convex relaxation constructed using McCormick's procedure is nonnegative on $\mathbb{X}$ and is flat on the convex hull defined by the four roots. The convex hull of the solution set S for the above system of equations and the solution set U of the associated McCormick convex program are shown in Fig. 2. U contains conv(S) with a close overlap as stated in Theorem 3. The automatically generated starting point (AGIG) $(-3, 3)$ obtained by solving the nonsmooth convex program is quite close to one of the solutions $(-3.77931, -3.283185)$ and RMT based damped-Newton iterations starting from the former easily converges to

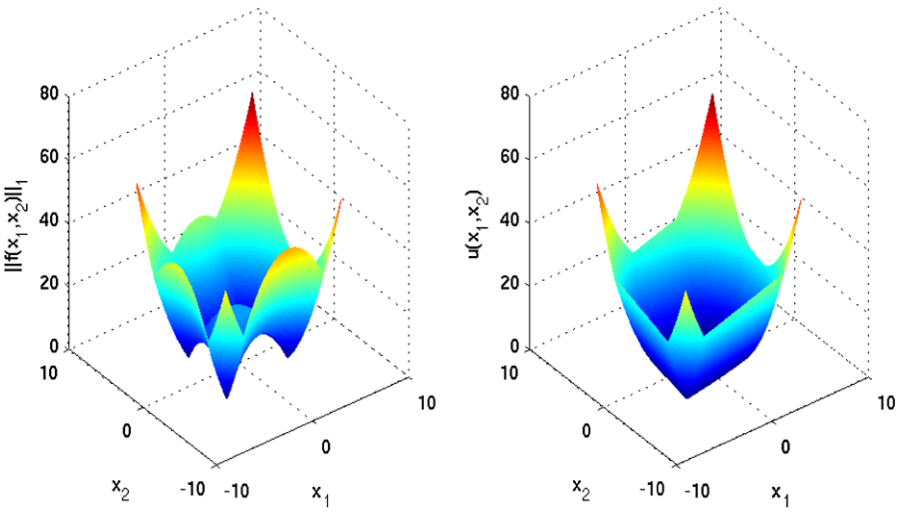

Fig. 1 Plot of $\|\mathbf{f}(\mathbf{x})\|_1$ (*left*) of Example 1 and its McCormick convex relaxation (*right*)

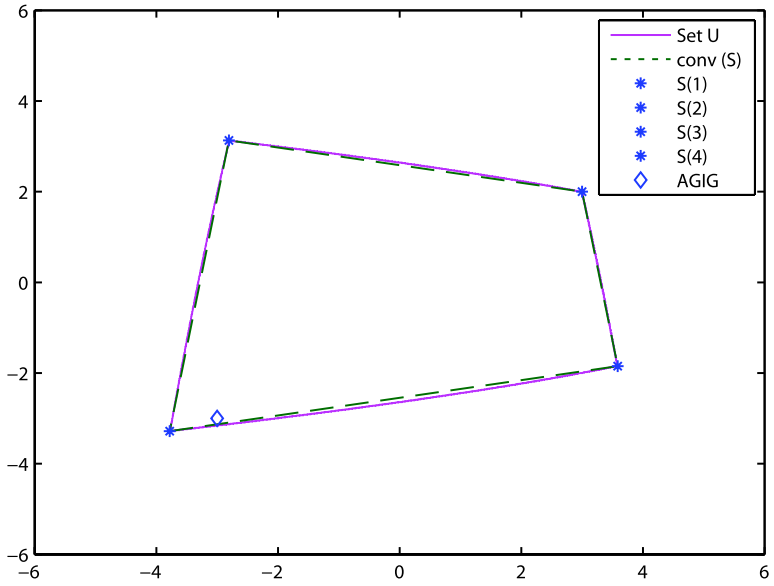

Fig. 2 Plot of conv (S) and set U corresponding to the system of equations in Example 1

the latter. All four roots of this problem were found in 45 iterations of the B&B algorithm with the componentwise Krawczyk operator implemented, and 39 iterations of the B&B algorithm with the interval-Newton with Gauss-Seidel implementation.

4 Branch-and-bound algorithm

The intuitive idea of the branch-and-bound algorithm is to search $\mathbb{X} \in \mathbb{IR}^n$ (defined by the variable bounds) exhaustively for solutions of (1) using a bisection strategy and to fathom portions of $\mathbb{X}$ based on certain criteria.

The algorithm starts with a stack N of nodes initialized to the given box $\mathbb{X}$ (in which the solutions of (1) are sought), the solution set S which is initially empty and the iteration counter k set to 1. Each node $\mathbb{X}_i \in N$ will have two associated attributes, namely, the solution field $\mathbb{X}_i.\mathbf{s}$ and the solution flag $\mathbb{X}_i.f$. By default, $\mathbb{X}_i.\mathbf{s}$ will be set to a point lying inside the node $\mathbb{X}_i$ (usually the mid point) and $\mathbb{X}_i.f$ is set to zero. If a solution has been found in the box $\mathbb{X}_i \in N$ it will be stored in its solution field $\mathbb{X}_i.\mathbf{s}$ when its solution flag $\mathbb{X}_i.f$ is set to one, indicating that a solution has been found in this node. The algorithm requires two tolerance parameters as input, namely (1) the size tolerance ϵ_{size} and (2) the feasibility tolerance ϵ_f. The size tolerance ϵ_{size} limits the smallest size of box to be examined, which essentially means that this algorithm can distinguish solutions which are ϵ_{size} apart in $\mathbb{R}^n$ in terms of the distance metric chosen to define the box size. Also, the nonsmooth solver will terminate at a certain tolerance and hence may not locate the exact zero value of the convex program even

if it exists. The feasibility tolerance ϵ_f limits the highest optimal value of the convex program below which the node will not be fathomed based on the root exclusion test.

At each iteration, a node is popped off the stack. Since the stack is a "last in, first out" (LIFO) container, the popped node will be the last one to be pushed in. If its size is smaller than ϵ_{size}, it is fathomed. Now, there are various ways of measuring the box size. The simplest one is the length of diagonal given by

$$d(\mathbb{X}) = \|\mathbf{x}^U - \mathbf{x}^L\|_2.$$

Another approach suggested in [28] is to use the scaled diameter of the box as a size metric defined as

$$d(\mathbb{X}) = \max_{1 \le i \le n} \{(x_i^U - x_i^L)/\max(|x_i^U|, |x_i^L|, 1)\}. \tag{17}$$

Using this size metric the algorithm may not be able to distinguish between roots which are ϵ_{size} apart in some dimension. One may decide based on the problem which size metric to choose.

If a node is not fathomed due to small size its solution flag is checked. If a solution has been found, the interval Newton-type root inclusion test is applied to check the uniqueness of the contained solution. If the inclusion test is positive the current box is fathomed. Otherwise, the box is bisected along a suitably chosen coordinate into two disjoint sub-boxes, such that the solution lies exactly in one of them (i.e., not on the boundary of the division). If the solution happens to lie on the bisection boundary, the bisecting plane is shifted either upwards or downwards along the bisected coordinate by a suitable amount (usually by a small chosen fraction of the interval width along the bisection coordinate of the box) to ensure that the solution is contained in exactly one of the sub-boxes. When a box $\mathbb{X} = (X_1, X_2, X_3, \ldots, X_n)$ is bisected, the resulting sub-boxes are $\mathbb{X}_L = (X_1, X_2, \ldots, [x_q^L, \overline{x}_q], \ldots, X_n)$ and $\mathbb{X}_U = (X_1, X_2, \ldots, [\overline{x}_q, x_q^U], \ldots, X_n)$, where $X_q = [x_q^L, x_q^U]$, $\overline{x}_q$ is the mid-point of the interval X_q and q is the coordinate chosen for bisection. The bisection coordinate can be chosen in two ways. A simpler and more intuitive way is to choose the coordinate with the widest interval. Another approach is to choose the direction having largest scaled diameter [28] such that q satisfies $d(X_q) = d(\mathbb{X})$, where $d(\mathbb{X})$ is defined according to (17). The latter scheme performs better (especially when the initial box widths vary widely in different coordinates) and has been used in the proposed algorithm.

To facilitate the branching of nodes, the algorithm uses a subroutine *Divide*. This subroutine takes as input a parent node $\mathbb{X}$ and returns two disjoint child nodes $\mathbb{X}_L$ and $\mathbb{X}_U$ obtained by division (usually bisection) of $\mathbb{X}$ such that the point in its solution field $\mathbb{X}.\mathbf{s}$ is contained in exactly one of them. It also sets the solution field and flag of the child nodes to their default values. The subroutine *Maxdim* returns the bisection coordinate of the parent interval vector $\mathbb{X}$ in q using a user defined size metric. In the pseudo code that follows it is assumed that equating any two nodes $\mathbb{Y}$ and $\mathbb{X}$ using $\mathbb{Y} = \mathbb{X}$ copies information stored in all the fields of $\mathbb{X}$ to the respective fields of $\mathbb{Y}$. Using these notations and those discussed at the beginning of this section, a pseudo

code for the subroutine *Divide* can be written as:

$$
\begin{aligned}
&[\mathbb{X}_L, \mathbb{X}_U] = \mathit{Divide}(\mathbb{X})\{ \\
&\ \mathbb{X}_L = \mathbb{X},\ \mathbb{X}_U = \mathbb{X},\ q = \mathit{Maxdim}(\mathbb{X}) \\
&\ \overline{x}_q = (x_q^L + x_q^U)/2 \\
&\ \mathbb{X}_L(q) = [x_q^L, \overline{x}_q],\ \mathbb{X}_U(q) = [\overline{x}_q, x_q^U] \\
&\ \text{if } (\mathbf{x}(q) = \overline{x}_q) \\
&\quad \eta = 0.1(x_q^U - x_q^L) \\
&\quad \mathbb{X}_L(q) = [x_q^L, \overline{x}_q + \eta],\ \mathbb{X}_U(q) = [\overline{x}_q + \eta, x_q^U] \\
&\quad \mathbb{X}_U.\mathbf{s} = \mathrm{mid}(\mathbb{X}_U),\ \mathbb{X}_U.f = 0 \\
&\ \text{end if} \\
&\}
\end{aligned}
$$

As per the above pseudo code, the point in the solution field of the parent node is contained in the lower sub-box $\mathbb{X}_L$. However, the choice of lower or upper sub-box for this purpose is arbitrary and one may choose either of the two sub-boxes to inherit the solution field of its parent.

Once bisected, the sub-box not containing the solution is pushed first, followed by the one which contains the solution and the iteration counter is increased by two to account for the newly generated two nodes. This ensures that in the next iteration the node containing the solution is again popped and this process will continue unless the solution containing node is fathomed either based on the inclusion test or the size metric. With the decreasing box size due to bisection at each iteration, the inclusion test will become more effective in the subsequent iterations and eventually the solution containing node will be fathomed based on the root inclusion test. Otherwise, even in the worst case it cannot escape the size-based fathoming, though after a much larger number of iterations.

If the current node does not contain a known solution, a simple interval-based root exclusion test is performed which is positive if the natural interval extension $\mathbb{F}$ of $\mathbf{f}$ over the current node $\mathbb{X}_i$ does not contain $\mathbf{0}$ and the node is fathomed. Otherwise, the interval Newton-type operator based interval root exclusion test, discussed in Sect. 3.3, is applied and if positive the current node is fathomed. If both these tests fail to fathom the current node $\mathbb{X}_i$, then the McCormick convex relaxation of $\|\mathbf{f}(\cdot)\|_1$ is constructed over $\mathbb{X}_i$ and the lower bounding convex program is solved using any nonsmooth solver (viz. PVARS [16]) and the obtained optimal point $\mathbf{x}^*$ is stored in its solution field. If the optimal value of the convex program is positive the current node is fathomed based on the nonsmooth root exclusion test. If the optimal value of the nonsmooth convex program is zero, then starting from $\mathbf{x}^*$ RMT based damped-Newton iterations are applied. The RMT solver is set to a convergence tolerance of ϵ_{Newton} and will converge if $\|\mathbf{f}(\mathbf{x}^j)\|_2 < \epsilon_{Newton}$ in a given maximum number of iterations. The RMT solver is said to fail if it does not converge to a solution in the given

maximum number of iterations, or the sequence of iterates crosses the box bounds before converging. If a solution is found the bisection process explained in the previous paragraph for a solution containing node is performed. Otherwise, the node is bisected by calling the subroutine *Divide* with the current node such that the automatically generated starting point $\mathbf{x}^*$ lies in exactly one of the two nodes obtained after the bisection. The resulting nodes are pushed onto the stack N with the one containing $\mathbf{x}^*$ being the last node to be pushed in and the iteration counter is increased by two.

This heuristic ensures that at any iteration of the algorithm, there will be only one, if any, solution containing node which lies at the top of the stack. Also, due to the bisection of nodes at each iteration, the McCormick convex relaxations become tighter and even "closer" starting points are obtained, resulting in quick convergence of the RMT based damped Newton method. As stated earlier, except for some of the test problems, a solution is obtained at the very first iteration of the algorithm and partial, if not full, credit to this does go to the generation of good starting points. Algorithm 1 formalizes the steps of the proposed branch-and-bound (B&B) algorithm for finding all real solutions of nonlinear system of equations.

Algorithm 1 (B&B algorithm for solving systems of nonlinear equations)

1. **(Initialization)**: Set $\mathbb{X}.f := 0, \mathbb{X}.\mathbf{s} = \text{mid}(\mathbb{X}), N = \{\mathbb{X}\}, S = \emptyset, k = 1$.
2. **(Termination)**: If $(N = \emptyset)$ then print the solution set S. Terminate.
3. **(Node Selection)**: Pop and delete the node $\mathbb{X}_i$ from the top of stack N.
4. **(Fathoming Based on Size)**: If $(d(\mathbb{X}_i) < \epsilon_{size})$ then goto 2.
5. **(Root Inclusion Test)**: If $(\mathbb{X}_i.f = 1)$ then [$\mathbb{X}_i$ contains a known solution]
   - $\bar{\mathbf{x}}^* := \mathbb{X}_i.\mathbf{s}$. Compute componentwise $\Phi(j)(\bar{\mathbf{x}}^*, \mathbf{f}, \mathbb{X}_i(j))$, where $\Phi(j) \in \{\mathbb{N}(j), \mathbb{K}(j)\}$.
   - If $(\Phi(j)(\bar{\mathbf{x}}^*, \mathbf{f}, \mathbb{X}_i(j)) \not\subset \mathbb{X}_i(j))$, set flag $\Rightarrow$ no inclusion. Increment j.
   - $\mathbb{X}_i(j) = \Phi(j)(\bar{\mathbf{x}}^*, \mathbf{f}, \mathbb{X}_i(j)) \cap \mathbb{X}_i(j)$.
   - If $j < n$ (size of system), repeat previous steps, else check inclusion flag. If no flag has been set for each j go to 2.
   - $[\mathbb{X}_k\ , \mathbb{X}_{k+1}] = \textit{Divide}(\mathbb{X}_i)$.
   - If $(\bar{\mathbf{x}}^* \in \mathbb{X}_k)$ then
     - Push $\mathbb{X}_{k+1}$ followed by $\mathbb{X}_k$ onto the stack N.
   - Else
     - Push $\mathbb{X}_k$ followed by $\mathbb{X}_{k+1}$ onto the stack N.
   - $k = k + 2$. Goto 2.
6. **(Root Exclusion Test)**: Compute the natural interval extension $F(\mathbb{X}_i)$ of $\mathbf{f}$ over $\mathbb{X}_i$.
   - If $(\mathbf{0} \notin F(\mathbb{X}_i))$ then goto 2 [$\mathbb{X}_i$ does not contain a solution].
   - Else
     - $\bar{\mathbf{x}}^* := \mathbb{X}_i.\mathbf{s}$. Compute componentwise $\Phi(j)(\bar{\mathbf{x}}^*, \mathbf{f}, \mathbb{X}_i(j))$, with $\Phi(j) \in \{\mathbb{N}(j), \mathbb{K}(j)\}$.

- If $(\Phi(j)(\bar{\mathbf{x}}^*, \mathbf{f}, \mathbb{X}_i(j)) \cap \mathbb{X}_i(j) = \emptyset)$ then goto 2. [$\mathbb{X}_i$ does not contain a solution].
- $\mathbb{X}_i(j) = \Phi(j)(\bar{\mathbf{x}}^*, \mathbf{f}, \mathbb{X}_i(j)) \cap \mathbb{X}_i(j)$, increment j and repeat previous 2 steps.
- Set $\mathbb{X}_i.\mathbf{s} = \text{mid}(\mathbb{X}_i)$.

7. **(Automatic Starting Point Generation)**: Construct the McCormick convex relaxation u of $\|\mathbf{f}(\cdot)\|_1$ over $\mathbb{X}_i$ and solve the resulting nonsmooth convex program using any nonsmooth solver. Let,
   - $\mathbf{x}^* \in \arg\min_{\mathbf{x}\in\mathbb{X}_i} u(\mathbf{x})$.
   - Set $\mathbb{X}_i.\mathbf{s} = \mathbf{x}^*$.
8. **(Nonsmooth Root Exclusion Test)**: If $(u(\mathbf{x}^*) > \epsilon_f)$ then goto 2 [$\mathbb{X}_i$ does not contain a solution].
9. **(RMT Based Damped-Newton Iterations)**: Apply a maximum of *maxiter* RMT iterations (NWTSLV) starting from $\mathbf{x}^* \in \mathbb{X}_i$. Let *niter* ($\leq$ *maxiter*) be the number of iterations taken by NWTSLV so that $\|\mathbf{f}(\bar{\mathbf{x}}^*)\|_2 \leq \epsilon_{Newton}$ where, $\bar{\mathbf{x}}^*$ stores the resulting solution.
   $[\bar{\mathbf{x}}^*, niter] = \text{NWTSLV}(\mathbf{x}^*, \mathbf{f}, \mathbb{X}_i, maxiter, \epsilon_{Newton})$.
   If $(niter \leq maxiter)$ [NWTSLV Converged] then set $\mathbb{X}_i.f = 1$ and $\mathbb{X}_i.\mathbf{s} = \bar{\mathbf{x}}^*$, $S = S \cup \{\bar{\mathbf{x}}^*\}$, goto 5.
10. **(Branching)**:
    - $[\mathbb{X}_k\ , \mathbb{X}_{k+1}] = Divide(\mathbb{X}_i)$.
    - If $(\mathbf{x}^* \in \mathbb{X}_k)$ then
      - First push $\mathbb{X}_{k+1}$ followed by $\mathbb{X}_k$ onto the stack N.
    - Else
      - First push $\mathbb{X}_k$ followed by $\mathbb{X}_{k+1}$ onto the stack N.
    - $k = k + 2$. Goto 2.

5 Computational results

In this section, a number of test problems from the literature are addressed to measure the efficacy of the proposed branch-and-bound algorithm in finding all solutions of systems of nonlinear equations. The computational times reported are on an Intel Core2 Quad (2.66 GHz) processor, allocating a single core and 512 MB of memory to the experiments. A size-tolerance of 10^{-4} and feasibility tolerance of 10^{-6} were used. For the RMT code NWTSLV parameters *maxiter* and ϵ_{Newton} were set to 20 and 10^{-8} respectively. For the nonsmooth solver (PVARS) the default tolerances were used [17]. The default tolerance on the decision variables and the function value is 10^{-8}. The performance of the algorithm is judged on the basis of the performance parameters shown in Table 1. The test problems from various sources are tabulated and explained in Table 2 and the performance of the algorithm on them is reported in Tables 3 and 4. A number of test problems (e.g., 3, 5, 8, 9, 12, 15, 30) contain transcendental functions. Some of the test problems (e.g., 27, 28, 29, etc.) can be decomposed into smaller blocks and can be very easily solved block-by-block.

Table 1 Performance parameters of the branch-and-bound algorithm

Parameter	Description
n	Space dimension
$\|S\|$	Cardinality of the solution set S
NIT	Branch-and-bound iterations before termination of the algorithm
NITF	Branch-and-bound iterations before the first solution is found
SZ	Number of nodes fathomed based on node size
INCLT	Number of nodes fathomed by the root inclusion test
EXT	Number of nodes fathomed by the root exclusion test
NSEXT	Number of nodes fathomed by nonsmooth root exclusion test
NWTF	Number of times RMT based damped-Newton method failed
MSD	Maximum stack depth reached prior to termination of the algorithm
CPU	CPU time taken by the algorithm in seconds

However, for testing the performance of the algorithm, they were not block decomposed at any stage to preserve the difficulty level. Also, to force the convergence in finite time, an upper limit of 50,000 on the maximum number of B&B iterations is imposed. The NIT values (number of iterations) for the problems on which the algorithm terminated without converging in 50,000 iterations are mentioned as NC (Not Converged) in the results table. As is evident from Tables 3 and 4, the first solution is found at the very first iteration of the algorithm for most of the test problems. This is due to the combined effect of the good starting point generation and the increased region of convergence of the damped-Newton method through the use of RMT with natural level function. Also, the efficacy of the proposed nonsmooth exclusion test is evidenced by the fact that it successfully fathoms nodes which are left unfathomed by the Krawczyk or interval-Newton root exclusion tests, since the former is performed only if the latter fails. The Krawczyk and interval-Newton operator based root inclusion tests also have an important contribution in reducing the number of B&B iterations, in lack of which, the only way to fathom the solution containing node is based on the size metric, leading to a significant increase in the number of iterations as well as processing time.

For problems 2, 5 and 6 the number of iterations are only a fraction of those reported in [18] using a similar branch-and-bound procedure with the αBB convex relaxation for generating lower bounds as against McCormick convex relaxation in the proposed method. For a fair comparison of the results using the two approaches, it should be noted that in the presented method, the iteration counts the actual number of nodes visited in the B&B tree, unlike in [18] where the iterations are increased by one (instead of two) at each bisection of the nodes.

For problem 7, the minimum value of α chosen for convex lower bounding the only nonconvex expression using the αBB procedure [18] should be 16, as dictated by the range of eigenvalues of the corresponding Hessian matrix. Hence, iteration counts reported for α less than 16 in [18] are not matched by the proposed algorithm. Nevertheless, the results are comparable with those in [18] for appropriate values of α.

Table 2 Test problems

Example	Brief description and sources
1	Global minima of the Himmelblau function (Example 1)
2	Stationary points of the Himmelblau function [18]
3	Multiple steady states of a CSTR reactor [11]
4	Production of synthesis gas in an adiabatic reactor [11]
5	Badly scaled system of equations [18]
6	Robot kinematic problem [18]
7	Brown's almost linear system [18]
8	Example problem 4 from [12]
9	Example problem 5 from [12]
10	Chemical equilibrium under a stoichiometric feed condition [5]
11	Chemical equilibrium under a non-stoichiometric feed condition [5]
12	Kinetics in a stirred reactor [5]
13	Adiabatic flame temperature computation [5]
14	Calculation of gas volume using the Beattie-Bridgeman equation [5]
15	Fractional conversion in a reactor [5]
16	Flow in a smooth pipe ($a = 240$, $b = 40$ and $c = 200$) [5]
17	Flow in a smooth pipe ($a = 46.64$, $b = 67.97$ and $c = 40$) [5]
18	Batch distillation at infinite reflux [5]
19	Volume from the virial equation of state [5]
20	Volume from the Redlich-Kwong equation of state [5]
21	Sinkage depth of a sphere in water [5]
22	Rosenbrock functions [24]
23	Freudenstein and Roth function [24]
24	Beale function [24] [case A (Eqs. 1 & 2), B (Eqs. 2 & 3) & C (Eqs. 1 & 3)]
25	Powell's singular function [24]
26	Wood's function [24]
27	Broyden tridiagonal function [24] [case A ($n = 2$), B ($n = 4$) & C ($n = 6$)]
28	Broyden banded function [24] [case A ($n = 2$), B ($n = 5$) & C ($n = 9$)]
29	Extended Rosenbrock function [24] [case A ($n = 10$), B ($n = 50$) & C ($n = 100$)]
30	Circuit design problem with extraordinary sensitivities to small perturbations [18]
31	Hydrocarbon combustion process [18]

For example 30, the algorithm failed to terminate after 50,000 iterations and no solution was found for either the Krawczyk of interval-Newton implementation. On increasing the *maxiter* for RMT solver NWTSLV from 20 to 100, no solutions were found and the method still failed to fathom the entire search space. A reason similar to that for problem 7 explains why the proposed method failed to converge in 50,000 iterations while in [18] it is reported to converge in 1645 iterations when $\alpha = 0.1$ is used for generating its αBB convex relaxation. It can be easily deduced by computing the natural interval extension of the Hessian of the last equation (which is the least nonconvex among the equations in the system) that the value of α should at least

Table 3 Performance of the Krawczyk B&B algorithm on the test problems

Ex.	n	$\|S\|$	NIT	NITF	SZ	INCLT	EXT	NSEXT	NWTF	MSD	CPU(s)
1	2	4	45	1	2	3	15	3	3	9	0.045
2	2	9	109	1	0	9	31	15	11	9	0.113
3	2	3	43	1	0	3	7	12	4	9	0.049
4	7	1	55	1	0	1	9	18	2	26	0.233
5	2	1	17	1	0	1	5	3	2	7	0.020
6	8	16	2239	1	38	0	431	651	60	88	3.783
7	5	2	513	1	14	1	94	148	141	83	1.199
8	1	2	39	1	0	2	6	12	5	11	0.038
9	1	5	137	1	0	5	11	53	33	10	0.140
10	1	1	7	1	0	1	3	0	0	4	0.011
11	1	1	7	1	0	1	0	3	1	3	0.011
12	1	1	3	1	0	1	1	0	0	2	6.5e–3
13	1	1	3	1	0	1	1	0	0	2	5.9e–3
14	1	2	25	1	0	2	9	2	2	8	0.017
15	2	1	9	7	0	1	3	1	3	5	0.025
16	2	1	13	1	0	1	5	1	1	6	0.013
17	2	1	21	1	0	1	7	3	1	10	0.028
18	1	2	35	19	0	2	14	2	15	12	0.053
19	1	1	21	1	0	1	2	8	3	8	0.026
20	1	1	15	1	0	1	3	4	0	8	0.020
21	1	2	13	1	0	2	4	1	0	5	0.017
22	2	1	3	1	0	1	1	0	0	2	0.016
23	2	1	51	23	0	1	16	9	20	6	0.066
24A	2	1	27	17	2	0	8	4	8	14	0.062
24B	2	1	35	3	2	0	12	4	3	16	0.035
24C	2	1	29	17	2	0	11	2	9	14	0.044
25	4	1	123	1	2	0	49	11	3	59	0.120
26	4	1	9	1	0	1	3	1	0	5	0.017
27A	2	2	27	1	0	2	2	10	3	6	0.039
27B	4	2	125	9	0	2	11	50	52	9	0.273
27C	6	2	489	47	0	2	38	205	229	16	1.140
28A	2	1	13	1	0	1	4	2	1	6	0.013
28B	5	1	35	1	0	1	6	11	6	12	0.065
28C	9	1	61	1	0	1	15	15	10	21	0.383
29A	10	1	3	1	0	1	1	0	0	2	0.013
29B	50	1	3	1	0	1	1	0	0	2	0.067
29C	100	1	3	1	0	1	1	0	0	2	0.474
30	9	1	NC	NC	NC	NC	NC	NC	NC	NC	436.5
31	5	1	NC	NC	NC	NC	NC	NC	NC	NC	217.7

Table 4 Performance of the interval-Newton B&B algorithm on the test problems

Ex.	n	$\|S\|$	NIT	NITF	SZ	INCLT	EXT	NSEXT	NWTF	MSD	CPU(s)
1	2	4	39	1	2	3	13	2	3	7	0.050
2	2	9	93	1	0	9	23	15	11	8	0.082
3	2	3	35	1	0	3	2	13	4	8	0.055
4	7	1	51	1	0	1	9	16	2	24	0.204
5	2	1	17	1	0	1	5	3	2	7	0.022
6	8	16	2113	1	32	0	381	644	45	83	3.627
7	5	2	499	1	14	1	91	144	138	83	0.755
8	1	2	25	1	0	2	4	7	2	9	0.018
9	1	5	127	1	0	5	4	55	33	9	0.135
10	1	1	3	1	0	1	1	0	0	2	4.9e–3
11	1	1	9	1	0	1	1	3	1	4	0.011
12	1	1	3	1	0	1	1	0	0	2	6.7e–3
13	1	1	3	1	0	1	1	0	0	2	5.3e–3
14	1	2	15	1	0	2	4	2	0	6	0.013
15	2	1	9	7	0	1	3	1	3	5	0.015
16	2	1	5	1	0	1	1	1	0	3	0.018
17	2	1	17	1	0	1	6	2	3	7	0.024
18	1	2	35	19	0	2	14	2	15	12	0.049
19	1	1	17	1	0	1	2	6	3	6	0.025
20	1	1	13	1	0	1	3	3	0	7	0.034
21	1	2	9	1	0	2	2	1	0	4	0.010
22	2	1	3	1	0	1	1	0	0	2	0.015
23	2	1	39	23	0	1	12	7	17	6	0.059
24A	2	1	25	17	2	0	7	4	8	13	0.038
24B	2	1	33	3	2	0	13	2	3	15	0.026
24C	2	1	27	17	2	0	10	2	9	13	0.042
25	4	1	123	1	2	0	49	11	3	59	0.118
26	4	1	9	1	0	1	3	1	0	5	0.013
27A	2	2	25	1	0	2	2	9	3	6	0.042
27B	4	2	121	9	0	2	12	47	52	9	0.252
27C	6	2	473	47	0	2	35	200	226	16	0.993
28A	2	1	11	1	0	1	3	2	1	5	0.023
28B	5	1	25	1	0	1	5	7	5	8	0.052
28C	9	1	43	1	0	1	11	10	8	14	0.305
29A	10	1	3	1	0	1	1	0	0	2	0.011
29B	50	1	3	1	0	1	1	0	0	2	0.072
29C	100	1	3	1	0	1	1	0	0	2	0.389
30	9	1	NC	NC	NC	NC	NC	NC	NC	NC	431.9
31	5	1	NC	NC	NC	NC	NC	NC	NC	NC	212.3

be 0.5. Furthermore, for the more nonconvex terms involving exponentials, the minimum value of α as calculated using the rigorous αBB criterion turns out to be of the order of 10^7 and if used will likely take many more iterations to converge than what has been reported. For this problem, the McCormick convex relaxation happens to be flat over a sizable part of the domain and even when the box-size becomes small, the relaxation does not becomes positive over the box. Because of the exponential terms and multiple occurrences of the same variable the natural interval extensions are very loose, resulting in the flat relaxation using McCormick's procedure.

Multiple occurrences of the same variable causes what is known as the dependency problem of natural interval extensions, leading to overly large bound estimation of the function range. Hence, for such functions, convex relaxations constructed using McCormick's method may be flat as in example 31 on which the algorithm failed to terminate after 50,000 B&B iterations for both the Krawczyk and interval-Newton implementations. As in example 30, the search space in this case also could not be fathomed fully with either implementation, primarily because of flat McCormick's convex relaxations on a large part of the domain.

In the proposed algorithm two root exclusion tests and one root inclusion test are operating simultaneously to extract the largest reduction of iteration count and hence the CPU time. However, to illustrate their individual effects, all the test problems were solved by switching off each of these two exclusion tests and the inclusion test. Each experiment was run utilizing a single interval Newton-type operator (Krawczyk or interval-Newton) so that the performance of each operator could also be compared. It should be noted that by deactivating the nonsmooth exclusion test, the classic interval Newton-type methods with generalized bisection are recovered. However, the one difference is that the solutions are found using the damped-Newton with RMT and reported to machine precision instead of being bounded by the interval Newton-type methods. Similarly, for comparison, the standard branch-and-bound method was recovered by deactivating the interval Newton-type inclusion/exclusion tests simultaneously. B&B iterations (NIT) and the CPU times taken for all the three cases are presented in Tables 5 and 6 respectively. In the following subsections the performance of the algorithm in each of these cases will be analyzed in detail.

5.1 B&B with Krawczyk operator vs. interval-Newton operator

When the interval-Newton operator with Gauss-Seidel is implemented for use in the interval Newton-type inclusion and exclusion tests, the number of nodes visited in the branch-and-bound tree is always less than that of the componentwise Krawczyk operator, except for a few examples. This is primarily due to the fact that the intervals generated by the interval-Newton operator with Gauss-Seidel are at least as tight as the componentwise Krawczyk, if not tighter, or in most cases, significantly tighter. For the examples where the number of nodes visited are equal or greater than the componentwise Krawczyk implementation, the interval-Newton operator is unbounded due to divisions by 0 ($0 \in A_{ii}$ for some i). If divisions by 0 persist, even upon further bisecting the current box, the interval-Newton inclusion test continues to fail, requiring more bisections. This is because uniqueness guarantees cannot be made when $0 \in A_{ii}$ for some i. Likewise, since the generated intervals are unbounded, the exclu-

Table 5 Performance of the B&B algorithm with one of its different features switched off

Ex.	NIT								
	Full Algorithm		EXT OFF		NSEXT OFF		INCLT OFF		Stnd B&B
	Krawczyk	I-Newton	Krawczyk	I-Newton	Krawczyk	I-Newton	Krawczyk	I-Newton	
1	45	39	45	39	113	105	285	285	285
2	109	93	109	93	199	159	593	593	593
3	43	35	43	35	87	137	219	219	279
4	55	51	55	51	39333	34139	223	223	223
5	17	17	17	17	253	227	73	73	81
6	2239	2113	2239	2113	NC	NC	4117	4117	4117
7	513	499	513	499	21425	30205	1269	1269	1269
8	39	25	43	25	69	65	57	49	63
9	137	127	137	127	263	965	229	229	229
10	7	3	7	3	7	3	27	27	27
11	7	9	7	9	13	19	29	29	29
12	3	3	3	3	3	3	41	41	41
13	3	3	3	3	3	3	51	51	51
14	25	15	33	15	33	21	51	51	61
15	9	9	9	9	7	7	57	57	57
16	13	5	13	5	17	7	61	61	63
17	21	17	21	17	37	27	59	59	59
18	35	35	35	35	51	51	63	63	63
19	21	17	21	17	41	69	51	51	51
20	15	13	15	13	25	21	29	29	37
21	13	9	13	9	21	11	63	63	63
22	3	3	3	3	3	3	67	67	67
23	51	39	51	39	113	103	123	123	123
24A	27	25	27	25	57	53	65	65	65
24B	35	33	35	33	51	47	69	69	69
24C	29	27	29	27	29	27	67	67	67
25	123	123	123	123	145	145	123	123	123
26	9	9	9	9	1017	1031	125	125	125
27A	27	25	27	25	61	59	135	135	135
27B	125	121	125	121	813	789	361	361	361
27C	489	473	489	473	9359	9105	843	843	843
28A	13	11	13	11	17	15	63	63	63
28B	35	25	35	25	399	187	173	173	173
28C	61	43	61	43	8349	3689	329	329	329
29A	3	3	3	3	3	3	707	707	1229
29B	3	3	3	3	3	3	NC	NC	NC
29C	3	3	3	3	3	3	20967	20967	NC
30	NC	NC	NC	NC	NC	NC	NC	NC	NC
31	NC	NC	NC	NC	NC	NC	NC	NC	NC

Table 6 Performance of the B&B algorithm with one of its different features switched off

Ex.	CPU (s)								
	Full Algorithm		EXT OFF		NSEXT OFF		INCLT OFF		Stnd B&B
	Krawczyk	I-Newton	Krawczyk	I-Newton	Krawczyk	I-Newton	Krawczyk	I-Newton	
1	0.045	0.050	0.039	0.044	0.122	0.107	0.130	0.151	0.048
2	0.113	0.082	0.095	0.079	0.171	0.147	0.315	0.314	0.090
3	0.049	0.055	0.047	0.054	0.070	0.147	0.119	0.113	0.101
4	0.233	0.204	0.259	0.224	61.13	52.26	0.430	0.436	0.661
5	0.020	0.022	0.019	0.020	0.255	0.231	0.050	0.040	0.043
6	3.783	3.627	3.717	3.422	78.50	77.94	5.500	5.360	2.161
7	1.199	0.755	0.992	0.890	29.18	44.59	1.321	1.368	0.746
8	0.038	0.018	0.041	0.028	0.068	0.057	0.038	0.031	0.030
9	0.140	0.135	0.139	0.133	0.243	1.117	0.157	0.154	0.073
10	0.011	4.9e–3	6.7e–3	4.7e–3	5.9e–3	5.6e–3	0.019	0.012	6.9e–3
11	0.011	0.011	8.2e–3	0.010	0.015	0.024	0.026	0.018	9.6e–3
12	6.5e–3	6.7e–3	5.4e–3	5.3e–3	4.2e–3	4.6e–3	0.029	0.018	9.8e–3
13	5.8e–3	5.3e–3	5.0e–3	5.3e–3	5.7e–3	4.9e–3	0.020	0.034	9.3e–3
14	0.017	0.013	0.030	0.011	0.032	0.019	0.049	0.023	0.019
15	0.025	0.015	0.018	0.014	8.7e–3	9.3e–3	0.035	0.040	0.023
16	0.013	0.018	0.011	0.017	0.025	0.010	0.040	0.030	0.030
17	0.028	0.024	0.026	0.016	0.031	0.034	0.039	0.038	0.025
18	0.053	0.049	0.048	0.049	0.053	0.058	0.052	0.057	0.048
19	0.026	0.025	0.017	0.018	0.032	0.082	0.031	0.046	0.023
20	0.020	0.034	0.012	0.011	0.052	0.045	0.099	0.050	0.017
21	0.017	0.010	0.011	9.2e–3	0.019	0.012	0.035	0.036	0.013
22	0.016	0.015	8.3e–3	5.3e–3	4.4e–3	5.2e–3	0.045	0.043	0.025
23	0.066	0.059	0.063	0.053	0.117	0.109	0.111	0.111	0.092
24A	0.062	0.038	0.038	0.034	0.062	0.045	0.040	0.043	0.031
24B	0.035	0.026	0.036	0.034	0.048	0.055	0.047	0.040	0.010
24C	0.044	0.042	0.031	0.042	0.024	0.017	0.057	0.047	0.051
25	0.120	0.118	0.121	0.108	0.117	0.137	0.093	0.099	0.078
26	0.017	0.013	0.015	0.013	1.391	1.266	0.075	0.095	0.060
27A	0.039	0.042	0.038	0.041	0.063	0.051	0.105	0.080	0.040
27B	0.273	0.252	0.253	0.250	0.870	0.846	0.411	0.427	0.227
27C	1.140	0.993	1.039	0.949	12.91	12.27	1.479	1.343	0.719
28A	0.013	0.023	0.012	0.011	0.045	0.019	0.035	0.123	0.019
28B	0.065	0.052	0.064	0.048	0.478	0.232	0.145	0.129	0.103
28C	0.383	0.305	0.335	0.219	15.68	7.466	0.716	0.814	0.650
29A	0.013	0.011	0.010	0.010	7.0e–3	7.1e–3	0.949	0.951	1.856
29B	0.067	0.072	0.054	0.055	0.096	0.067	1248	1183	1795
29C	0.474	0.389	0.538	0.541	0.236	0.223	1385	1387	8970
30	436.5	431.9	394.5	388.4	213.6	208.4	482.5	477.1	199.2
31	217.7	212.3	212.5	208.8	331.7	330.1	390.8	264.0	61.47

sion test cannot be utilized since $\mathbb{N}$ and $\mathbb{X}$ will not be disjoint. This is the potential situation when considering problems with multiple roots that are very close to one another or problems that have a singular Jacobian for most of the search space as in Example 11. Fathoming the entire search space for these types of problems requires searching very small subsets of the entire space, thus requiring more iterations in the branch-and-bound algorithm.

Since each considered interval Newton-type operator requires similar computational effort, it is expected that the solution times will also be lower for the interval-Newton operator with Gauss-Seidel. This is indeed the case, again with a few exceptions. It can be concluded that although there are a few examples where $0 \in A_{ii}$ for some i for narrow A_{ii}, the interval-Newton with Gauss-Seidel is a superior interval Newton-type operator as compared with the componentwise Krawczyk. Likewise, it may be useful to switch between the two methods on-the-fly when divisions by zero are detected.

5.2 B&B without interval Newton-type root exclusion tests

When the interval Newton-type operator root exclusion test is switched off, the iteration count remains the same for all of the problems except for Examples 8 and 14 with the Krawczyk operator, where it is increased only slightly. This means that most of the nodes which were fathomed by the interval Newton-type exclusion tests are also successfully fathomed by the proposed nonsmooth exclusion test. For the mentioned problems, there is only a small increase in the number of iterations because the nonsmooth exclusion test was not positive for some of the nodes which were earlier filtered by the Krawczyk exclusion test. Hence, subsequent bisections are required to have a tight enough convex relaxation before successful fathoming by the nonsmooth exclusion test. Nevertheless, a small increase in the iterations indicates that such nodes are small in number and usually the convex relaxation on the nodes was tight enough not to require too many subsequent bisections to be fathomed by the proposed exclusion test.

Barring a few exceptions, the CPU times taken by the algorithm decreases on switching off the interval Newton-type exclusion test. If the nonsmooth solver takes a larger number of iterations to solve the lower bounding convex program it will overwhelm the time saved in avoiding the computation of interval Newton-type operator. As a result, the overall CPU time will increase and this increment will further depend on the number of nodes on which the convex program is solved. In most of the presented test cases, as the node count (B&B iterations) does not increase by switching off the interval Newton-type exclusion test, it can be inferred that the additional number of nonsmooth convex program solved were nearly the same as the number of nodes fathomed by the interval Newton-type exclusion test when it was turned on. Hence, for most practical cases, the proposed nonsmooth exclusion test is as effective as the Krawczyk and interval-Newton exclusion tests.

5.3 B&B without nonsmooth root exclusion test

Turning off the nonsmooth root exclusion test, on the other hand, generally leads to a dramatic increase in the number of B&B iterations as well as the CPU time (except

for examples 30 and 31 for which the algorithm did not converge in 50,000 iterations). If NSEXT denotes the number of nodes fathomed by the nonsmooth exclusion test when all features of the algorithm were turned on, then upon turning it off, the increase in the B&B iterations (NIT) will tend to be at least 2×NSEXT. However, the observed increment in NIT is much more than this for most of the test problems. This demonstrates the prowess of the proposed nonsmooth root exclusion test as compared to the interval Newton-type exclusion tests. The increase in NIT is especially significant in problems of higher dimension and/or those with higher cardinality of the solution set S (e.g., 4, 5, 6, 7, 9, 27 and 28). In fact, without the nonsmooth exclusion test, for the robot kinematic problem 6 having as many as 16 solutions in a small volume of $[-1, 1]^8$, the whole search space could not be fathomed and the algorithm terminated when the maximum limit of 50,000 on NIT is achieved, only fathoming about half of the search space.

Consider those NSEXT nodes that were fathomed by the nonsmooth exclusion test. With the latter being turned off, all these nodes will now filter down to that stage of the B&B algorithm where RMT iterations are started, escaping the interval Newton-type exclusion tests. As no solution is contained in those nodes the RMT iterations will eventually fail to find a solution and will end up only in increasing the processing time. Furthermore, since the convex lower bounding problem is no longer solved, the starting points are also not generated automatically for the RMT iterations. Hence, in this case, the RMT iterations are started from the mid-point of the box which also leads to more failures of NWTSLV for some of the test problems. These increased failure of the RMT iteration (NWTSLV), leads to a dramatic increase in the CPU time for most of the test cases. However, for some problems (e.g., 12, 13, 22 and 29) there is no change in the iteration count because the nonsmooth exclusion test was not performed on any of the nodes and so switching it off does not effect the performance parameters.

With the nonsmooth exclusion test switched off, the remaining steps of the B&B algorithm closely approximates the interval Newton-type/bisection method for solving nonlinear equations but with the difference that RMT iterations are also applied on the boxes. Thus, even if it takes too many iterations to fathom the search space, solutions, if any, were found robustly and rapidly. As emphasized earlier, an added advantage of having the solution known a priori is excellent preconditioning of the interval Newton-type operators, which makes the root inclusion test quite effective. Moreover, the intersection relation in the interval Newton-type based exclusion and inclusion tests leads to better refinement of the regions where the solutions are likely to lie.

5.4 B&B without interval Newton-type root inclusion tests

When the root inclusion test is switched off, there is a significant increase in the B&B iterations for both the Krawczyk and interval-Newton root inclusion tests. However, the increment is fixed depending upon the dimensionality n of the problem, the cardinality of its solution set S and last, but not the least, on the initial box-size. If a solution is found in a box, then with the inclusion test switched off, the box can only be fathomed when its size becomes smaller than the threshold box-size limit (ϵ_{size})

by subsequent bisection. This leads to the generation of an exponential number of nodes and so the iteration count will increase dramatically. The increase in iteration count will also depend on how large a box the inclusion test fathomed when it was switched on. For example, in problems 24 and 25, the nodes in which a solution was found was fathomed based on the size metric and not by the inclusion test even when the test was turned on. Hence, one may expect no change in the number of iterations when the inclusion test is turned off. But, the test also helps in reducing the box size by intersecting it with the interval Newton-type operator as discussed in Sect. 3.3 which explains the increment in NIT for these problems.

The CPU time shows an increase for all the test problems. The nodes generated by bisection of the solution containing nodes are not easily fathomed by the exclusion tests, especially when the solution lies near the boundary of bisection. Hence, such nodes will experience the relatively expensive computational steps of the algorithms such as calculation of the Krawczyk or interval-Newton operator, solution of nonsmooth convex program and even the RMT iterations (in case both these tests fail to fathom the node). Hence, the increase in the processing time will depend on the number of nodes generated and the amount of computationally expensive steps that each such node undergoes. The increase in B&B iterations as well as the CPU time is particularly significant in the solution of the *extended Rosenbrock function* (Examples 29 A, B and C) where the problems are relatively bigger in size and the inclusion test, when on, was able to fathom a large volume of the search space.

From the numerical experiments outlined above, it seems that the combined effect of all of the inclusion and exclusion tests working together is considerably better than any of them used in isolation, both in terms of iteration counts and processing time. Also, the obtained test results also illustrate that the proposed nonsmooth exclusion test outperforms the interval Newton-type operator based exclusion tests when used in isolation. However, their combined effect yields the best performance and is better than either of the two acting independently. Furthermore, for most of the problems from the literature, the interval-Newton with Gauss-Seidel operator was the superior interval method as compared to the componentwise Krawczyk method, as the B&B algorithm converged in less iterations and less time. Therefore, it should be considered first before the componentwise Krawczyk, or alternatively, automatically switch from the interval-Newton to the Krawczyk when divisions by zero are detected.

5.5 Standard B&B

Simultaneously deactivating the interval Newton-type inclusion/exclusion tests recovers the standard B&B algorithm with McCormick relaxations. The standard B&B algorithm with McCormick relaxations is a novel approach to the root-finding problem, and to our best knowledge has not been implemented previously in the literature. In comparison to the full algorithm, the number of iterations taken to solve the test problems is significantly higher. On the other hand, since the algorithm is not utilizing the interval Newton-type inclusion/exclusion tests, there is a sizable reduction in operations performed per iteration. Therefore, the standard B&B algorithm with McCormick relaxations handles most of the test problems quite efficiently. For many of the examples the solution times are comparable or slightly better than the full algorithm, but with some exceptions with very long run times. On the other hand, the

full algorithm, with all of the inclusion/exclusion tests enabled, offers near the lowest run time for all the test problems, exhibiting the best average performance. It is clear from this comparison that although the standard B&B algorithm is very efficient and handles most of the test problems extremely well, the full algorithm obviously benefits from the ability to fathom the search space efficiently by utilizing all the inclusion/exclusion tests available, and the increased cost per iteration is outweighed by the best average performance over the test set.

6 Conclusion and future work

McCormick's convex relaxations seem very promising for convex lower bounding of nonconvex programs because of their capability to produce relatively tight relaxations. This is particularly helpful in the solution of systems of nonlinear equations using a global optimization approach as investigated in this paper. In this formulation, due to the special structure of the objective function, nonnegativity of the McCormick convex relaxation can be established leading to a root exclusion test. This nonsmooth root exclusion test has a significant edge in performance over the interval Newton-type root exclusion tests as demonstrated by a number of test problems discussed in Sect. 5. Another important contribution is the set inclusion relation asserted by Theorem 3 providing a technique for automatic generation of starting points for point Newton-type iterations.

A distinguishing feature of the proposed algorithm is that by embedding the RMT based Newton-type method in a branch-and-bound framework, a solution can be located at the very first iteration for most of the test problems. This is further exploited to fathom the search space more efficiently using the Krawczyk or interval-Newton root inclusion test by checking the uniqueness of the known solution in the given box.

Complete global convergence is guaranteed by the design of the algorithm itself. The bisection strategy combined with various inclusion and exclusion tests and the allowed minimum box-size limit ensures that the algorithm will converge finitely. However, the key question is the high computational effort required for large problems where special concerns arise for solving the nonsmooth convex program and the amount of branching and partitioning required. The efficiency of the algorithm also depends on the range specification of the variables. Overly large bounds on variables not only lead to weaker interval extensions and bad relaxations, but also make the interval based inclusion and exclusion tests ineffective until sufficient contraction on variable bounds is achieved by successive bisection.

An important scope for future work is motivated by the convexity of the set U. As established earlier that, for zero optimal value of the lower bounding convex program, the solution set U is a convex set which contains all solutions of (1). Hence it would be desirable to prevent the point Newton-type iterations from leaving the set U. This can be done utilizing the convexity of U and using the separating hyperplane theorem. If the Newton-type iterate generates a point outside U then a separating hyperplane which separates this point from U can be constructed by solving another convex program and it can be imposed as a constraint or "cutting plane" to further restrict the admissible domain. This will potentially cut down the iteration

counts for Newton-type methods significantly though at the expense of the involved computational efforts.

Acknowledgements The authors would like to acknowledge the Chevron University Partnership Program for supporting this work through the MIT Energy Initiative. This research was carried out as a part of Masters' thesis of the second author as a graduate student in the Computation for Design and Optimization Program at MIT, USA. The second author is grateful to the Singapore-MIT Alliance for funding his graduate studies and this project thereof at MIT.

References

1. Bock, H.G., Kostina, E., Schloder, J.P.: On the role of natural level functions to achieve global convergence for damped-Newton method. In: System Modelling and Optimization: Methods, Theory and Applications, pp. 51–74 (2000)
2. Chachuat, B.: libMC: A numeric library for McCormick relaxation of factorable functions (2007). http://yoric.mit.edu/libMC/
3. Deuflhard, P.: A modified Newton method for the solution of ill-conditioned systems of equations with applications to multiple shooting. Numer. Math. **22**, 289–315 (1974)
4. Griewank, A.: Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation. SIAM, Philadelphia (1987)
5. Gritton, K.S., Seader, J.D., Lin, W.J.: Global homotopy continuation procedures for seeking all roots of a nonlinear equation. Comput. Chem. Eng. **25**, 1003–1019 (2001)
6. Hansen, E., Sengupta, S.: Bounding solutions of systems of equations using interval analysis. BIT Numer. Math. **21**, 203–211 (1981)
7. Kearfott, R.B.: Abstract generalized bisection and a cost bound. Math. Comput. **49**(179), 187–202 (1987)
8. Kearfott, R.B.: Preconditioners for the interval Gauss-Seidel method. SIAM J. Numer. Anal. **27**(3), 804–822 (1990)
9. Kearfott, R.B.: Rigorous Global Search: Continuous Problems. Kluwer Academic, Boston (1996)
10. Kearfott, R.B., Novoa, M.: INTBIS, a portable interval-Newton/bisection package. ACM Trans. Math. Softw. **16**, 152 (1990)
11. Kuno, M., Seader, J.D.: Computing all real solutions to systems of nonlinear equations with global fixed-point homotopy. Ind. Eng. Chem. Res. **27**, 1320–1329 (1988)
12. Liang, H., Stadtherr, M.A.: Computation of interval extensions using Berz-Taylor polynomial models. In: AIChE Annual Meeting, Los Angeles, CA (2000)
13. Lucia, A., Feng, Y.: Global terrain methods. Comput. Chem. Eng. **26**, 529–546 (2002)
14. Luksan, L., Vlcek, J.: A bundle-Newton method for nonsmooth unconstrained minimization. Math. Program. **83**(3), 373–391 (1998)
15. Luksan, L., Vlcek, J.: Globally convergent variable metric method for convex nonsmooth unconstrained minimization. J. Optim. Theory Appl. **102**(3), 593–613 (1999)
16. Luksan, L., Vlcek, J.: Algorithm for non-differentiable optimization. ACM Trans. Math. Softw. **2**(2), 193–213 (2001)
17. Makela, M.M.: Survey of bundle methods for nonsmooth optimization. Optim. Methods Softw. **17**(1), 1–29 (2002)
18. Maranas, C.D., Floudas, C.A.: Finding all solutions of nonlinearly constrained systems of equations. J. Glob. Optim. **7**(2), 143–182 (1995)
19. McCormick, G.P.: Computability of global solutions to factorable nonconvex programs, part I: convex underestimating problems. Math. Program. **10**, 147–175 (1976)
20. Mitsos, A., Chachuat, B., Barton, P.I.: McCormick-based relaxations of algorithms. SIAM J. Optim. **20**(2), 573–601 (2009)
21. Moore, R.E.: A test for existence of solutions to nonlinear systems. SIAM J. Numer. Anal. **14**(4), 611–615 (1977)
22. Moore, R.E.: Methods and Applications of Interval Analysis. SIAM, Philadelphia (1979)
23. Moore, R.E., Kioustelidis, J.B.: A simple test for accuracy of approximate solutions to nonlinear (or linear) systems. SIAM J. Numer. Anal. **17**(4), 521–529 (1980)

24. Moré, J.J., Garbow, B.S., Hillstrom, K.E.: Testing unconstrained optimization software. ACM Trans. Math. Softw. **7**(1), 17–41 (1981)
25. Neumaier, A.: Interval Methods for Systems of Equations. Cambridge University Press, Cambridge (1990)
26. Neumaier, A.: Complete search in continuous global optimization and constraint satisfaction. Acta Numer. **13**, 271–369 (2004)
27. Ortega, J.M., Rheinboldt, W.C.: Iterative Solution of Nonlinear Equations in Several Variables. Academic Press, San Diego (1970)
28. Schnepper, C.A., Stadtherr, M.A.: Robust process simulation using interval methods. Comput. Chem. Eng. **20**, 187–199 (1996)
29. Wilhelm, C.E., Swaney, R.E.: Robust solution of algebraic process modelling equations. Comput. Chem. Eng. **18**(6), 511–531 (1994)