INTERVAL ALGORITHMS FOR FINDING THE MINIMAL ROOT IN A SET OF MULTIEXTREMAL ONE-DIMENSIONAL NON-DIFFERENTIABLE FUNCTIONS

L.G. CASADO[†§], I. GARCíA[‡§], AND YA.D. SERGEYEV[¶]

Abstract. Two problems arising very often in applications are considered. The first problem consists of finding the minimal root of an analytic one-dimensional function over a given interval. It is supposed that the objective function can be multiextremal and non-differentiable. Three new algorithms based on Interval Analysis and Branch-and-Bound Global Optimization approaches are proposed for solving this problem. The novelty of the new algorithms is in improving the elimination criteria and the order in which interval and point evaluations are realized. The techniques introduced accelerate the search in comparison with interval analysis methods traditionally used for finding roots of equations. The second problem considered in the paper is a generalization of the first one and deals with the search for the minimal root in a set of multiextremal and non-differentiable functions. The methods proposed for solving the first problem are generalized for solving the second. The main idea is to use the information obtained from any of the functions to reduce the search domain associated with all the functions. Numerical experiments carried out on a wide set of test functions demonstrate a quite satisfactory performance of the new algorithms in both cases.

Key words. Minimal Root, Interval Analysis, Branch-and-Bound.

AMS subject classifications. 65-04,65-05,65G10,65H20,65K05,90C30

1. Introduction. Many applications give rise to the problem of finding the roots of one-dimensional functions subject to box constraints. For example, in the computer graphics and visualization field, one of the fundamental operations in ray tracing consists of the calculation of intersections between rays and objects [3, 11, 17]. Other examples of such fields and applications are: (i) digital signal processing techniques, i.e. in the non-parametric identification of special signals [24], in the wavelet transform [16], and in image processing [5], (ii) matrix computation for the eigenvalue problem [1], (iii) the design of electrical and electronic simulation software tools for time-domain and switching networks analysis [2].

In this paper two problems arising from such applications are considered. The first one consists of providing a general, simple, reliable, and fast algorithm for finding the minimal root of an analytically defined one-dimensional (possibly multiextremal) function. The second problem is a generalization of the first one and deals with finding the minimal root in a set of analytically defined one-dimensional functions. Let us describe the first problem.

Given a real one-dimensional function $f : S \subset I\!R \to I\!R;\ S = [a, b],\ a \leq b$, where S is the search domain of the problem, it is necessary to determine

$$s^* = \min\{s \ : \ f(s) = 0,\ s \in S\} \tag{1.1}$$

or

$$\begin{array}{l} s_u = \min\{s \ : \ f(s) \cdot f(a) \leq 0,\ s \in S\}, \text{ followed by} \\ s_l = \max\{s \ : \ f(s) \cdot f(a) \geq 0,\ s < s_u\} \end{array} \tag{1.2}$$

[†]E-mail: leo@ace.ual.es

[‡]E-mail: inma@ace.ual.es

[§]Department of Computer Architecture and Electronics, University of Almería, Spain. This work was supported by the Ministerio de Educación y Cultura of Spain (CICYT TIC99-0361).

[¶]E-mail: yaro@si.deis.unical.it. ISI-CNR c/o DEIS, Universitá della Calabria, 87036 Rende (CS) Italy, and University of Nizhni Novgorod, Nizhni Novgorod, Russia.

Most of the algorithms previously proposed were not specifically designed for finding the first root but all the roots. Nevertheless, there exists a wide set of applications (see [1, 2, 3, 5, 11, 16, 17, 24]) where only the minimal root contains valuable information for the problem at hand, so it is advisable to devise specific algorithms for finding only the first root.

Several approaches were proposed for solving this problem. The simplest approach is based on the use of grid techniques which produces a dense mesh starting from the left margin of the interval and progressing by ϵ, until two consecutive values of the function ($f(x)$ and $f(x+\epsilon)$) have different signs [2]. For small values of ϵ, this approach is very (but not totally) reliable and computationally very expensive.

The second approach consists of using standard local techniques which achieve a rapid convergence to a root. The main drawback of these methods is that convergence is not assured when $f(x)$ is a multiextremal function on S [21]. Moreover, if the objective function $f(x)$ has several roots, different choices of the initial conditions can lead to different solutions for the equation $f(x) = 0$. A standard and fast local search technique is the well known Newton method, which frequently provides a quadratic convergence. Nevertheless, Newton method may also fail, depending on the initial point. For instance, Newton method does not converge for the function $f(x) = x^3 - x$, if the initial point is $x = \frac{1}{\sqrt{5}}$.

Mitchell's proposal [17] is based on an interval algorithm introduced by Moore [19]. It distinguishes two stages: (i) the root isolation stage consists of finding the set of subintervals which are known to contain one and only one root, and (ii) the root refinement stage which uses a standard and fast local search technique, such as Newton method. Caprani et. al. [3] improved Mitchell's algorithm by using interval methods in both stages (isolation and refinement). They demonstrated that using interval analysis in both stages does not increase the execution time, as was stated by Mitchell. One of the advantages of interval methods is that they can provide information on the local uniqueness of the roots at the solution intervals [3, 9, 19, 20]. Interval methods guarantee that an interval is rejected only when it does not contain a root.

The third class of approaches, proposed by Daponte et al. [6, 7, 23] and Kalra and Barr [11], works on the framework of Global Optimization techniques based on the computation of Lipschitzian derivatives. In [6, 7, 23] several efficient solutions have been proposed for solving problems in the electrical engineering field. The paper of Kalra and Barr [11] deals with applications from the visualization and computer graphics field.

The fourth approach to solve (1.2) consists of using existing Nonlinear Global Optimization algorithms based on the Branch-and-Bound strategy and Interval Arithmetic, as those proposed in [8, 9, 12, 13, 22].

The computational model of the algorithms developed in this paper is based on interval arithmetic computations [17, 19] and also incorporates the ideas described in [6, 7, 23]. The new algorithms are specifically designed for finding the minimal root and do not need to determine Lipschitz constants or to evaluate derivatives. These algorithms develop and improve the ideas presented in [4].

The second problem considered in this paper is a generalization of the problem (1.1). There exist several applications where is necessary to find the minimal root in a set of p functions instead of a single function; i.e. to find the minimum of the minimal root of several functions [7, 11, 17]. An example of these applications appears in computer graphics where the scene to visualize is composed of a set of objects defined

by their implicit surfaces. Calculation of intersections between rays and primitive solids or surfaces is one of the fundamental operations to carry out. Visualizing a scene implies that for every ray only one of the intersections between that ray and all the objects is needed (the closest to the observation point) [11, 17]. Mathematically, the problem can be posed as follows:

Given a ray, represented parametrically by a starting point $a=(a_x, a_y, a_z)$, a direction vector V and a set of p implicit surfaces, $f_i(x,y,z) = f_i(a_x+tV_x, a_y+tV_y, a_z+tV_z)=f_i(t)$; i.e. $f_i : t \in R \to R$; $t \in S=[\underline{s}=0, \overline{s}]$, $i = 1, \ldots, p$, to determine:

$$t^* = \min\{t \in S : f_i(t) = 0; \quad \forall i,\ i = 1, \ldots, p\} \tag{1.3}$$

In the second part of this paper, the set of methods proposed to deal with problems described by (1.1) is generalized for solving (1.3).

The rest of the paper was organized as follows: Section 2 presents a general scheme of the interval algorithms which can be used for solving the problem (1.1), and three methods belonging to this scheme are described. In Section 3, in order to evaluate the performance of the methods proposed in this paper numerical results obtained from experiments made on a set of forty test functions are given. In Section 4, algorithms described in Section 2 are generalized to solve the problem of finding the minimal root in a set of one-dimensional functions. Numerical tests for the generalized methods are presented and discussed in Section 5. Finally, Section 6 briefly describes the main conclusions of this work.

2. Algorithms for finding the minimal root of a single multiextremal non-differentiable function over an interval. In order to proceed with the description of the algorithms, let us introduce the following terminology. Hereinafter, real numbers are denoted by $x, y, \ldots$, and real bounded and closed one-dimensional intervals by $X, Y, \ldots$, where $X = [\underline{x}, \overline{x}] = \{x : \underline{x} \leq x \leq \overline{x}\}$, and $\underline{x}$ and $\overline{x}$ are the lower and upper bounds of the interval X, respectively. The set of compact intervals is denoted by $\mathbb{I} := \{[a,b] : a \leq b,\ a, b \in \mathbb{R}\}$. The width and the midpoint of an interval X are defined by $w(X) = \overline{x} - \underline{x}$ and $m(X) = (\underline{x}+\overline{x})/2$, respectively. When we express a real number as an interval, we shall usually retain the simpler noninterval notation, i.e, x instead of $[x, x]$.

The scheme introduced here for solving problem (1.1) falls into the general framework of Branch-and-Bound (B&B) algorithms. The basic idea in B&B methods consists of a recursive decomposition of the original problem into smaller disjoint subproblems until the solution is found. The search avoids visiting those subproblems which are known not to contain a solution. B&B methods can be characterized by four rules: *Branching, Selection, Bounding*, and *Elimination* [10, 18]. For problems where the solution is determined when a desired accuracy is reached, a *Termination rule* has to be incorporated.

The general scheme for describing interval B&B methods can be summarized by the following rules:

Branching rule: The initial search region is denoted by $S = [\underline{s}, \overline{s}]$. The *branching rule* divides the selected interval $X \subseteq S$ using the traditional bisection method. The generated subintervals (we shall call these "active" intervals) are stored in a work list in a non-increasing order based on $\underline{x}$, i.e, the subinterval with greater $\underline{x}$ is stored first.

Selection rule: The next interval to be processed is the one at the head of the work list. Therefore, the algorithm always works on the left most active interval.

Bounding rule: It is based on the evaluation of an *inclusion function*, $F(X)$, at the interval $X = [\underline{x}, \overline{x}] \subseteq S$. $F : \mathbb{I} \to \mathbb{I}$ is an *inclusion function* of $f : \mathbb{R} \to \mathbb{R}$, if $f(X) \subseteq F(X)$, where $f(X)$ is the range of f on X. In the present study the *inclusion function* of the objective function is obtained by natural interval extension [14, 19, 22]. However, centered forms, slope forms [12] or the Taylor models [15] are also feasible tools for computing inclusion functions.

Elimination rule: Intervals which do not contain the minimal root are determined in some way. These intervals are rejected.

Termination rule: Intervals $X : w(X) \leq \epsilon$, which were selected and not rejected, are stored in the final list instead of in the work list. The input parameter ϵ determines the desired accuracy of the problem solution. The algorithm finishes when the work list is empty.

Three algorithms fitting in the above scheme are discussed (see Algorithms 1–3). The following notation is used in their description:

S: The search region $S = [\underline{s}, \overline{s}]$
F: The natural interval extension of f.
L: Priority Working List.
Q: Final List containing a set of intervals $X \subseteq S : w(X) \leq \epsilon$ and $0 \in F(X)$.
ϵ: Termination criterion. If $w(X) \leq \epsilon$, X is stored in Q.
$+/-$: Denote including/discarding boxes in a list.

Algorithm 1 (Roots Finder (RF)) is similar to Mitchell's algorithm. Their main difference is that RF does not compute derivatives, while Mitchell's algorithm uses derivatives to determine that there is only one root in X when $0 \notin F'(X)$ and $\overline{[F(\overline{x}) \cdot F(\underline{x})]} \leq 0$. The RF algorithm only uses the traditional elimination rule; i.e. it rejects intervals $X \subseteq S : 0 \notin F(X)$, because $0 \notin f(X)$ is assured. In numerical comparisons, RF will be used as a reference to measure the efficiency of the new methods. The set of intervals returned by Algorithm 1 is an enclosure of all the roots of f in S. However, due to the overestimations introduced by Computational Interval Arithmetic, in general, from $0 \in F(X)$ it cannot be concluded that $0 \in f(X)$. As an example, Figure 2.1 shows a zoom of the graphical output generated by the execution of Algorithm 1 for the problem number 20, (see Table 3.1).

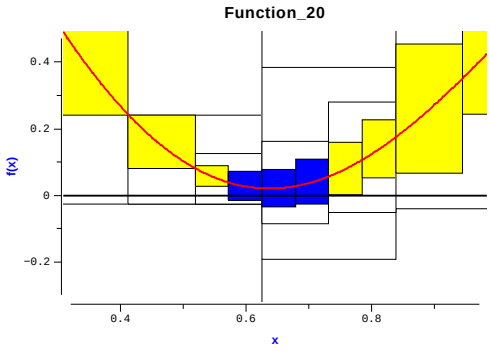


FIG. 2.1. *Zoom of the graphical output generated by Algorithm 1 for the problem number 20 using* $\epsilon = 10^{-2} \cdot w(X)$

In Figures 2.1-2.3, evaluated intervals, X, are denoted by a box bounded by

Algorithm 1 : Roots Finder Algorithm

Funct RF(S, F, ϵ, L, Q)

1. $Q := \{\}$; $L := \{S\}$ — *Initializing Final and Work Lists*
2. **while** ($L \neq \{\}$)
3. $X :=$ Head(L)
4. $L := L - \{X\}$
5. **if** $(0 \in F(X))$ — *Traditional interval rejection rule*
6. **if** $(w(X) \leq \epsilon)$
7. $Q := Q + \{X\}$ — *X belongs to the final solution*
8. **else**
9. Bisect (X, X_l, X_r) — $X_l = [\underline{x}, m(X)]$, $X_r = [m(X), \overline{x}]$
10. $L := L + \{X_r\} + \{X_l\}$ — *Subintervals are stored in Head(L)*
11. **return** Q

Algorithm 2 : The Minimal Root Finder Algorithm

Funct MRF (S, F, ϵ, L, Q)

1. $Q := \{\}$; $L := \{S\}$ — *Initializing Final and Work Lists*
2. **while** ($L \neq \{\}$)
3. $X :=$ Head(L)
4. $L := L - \{X\}$
5. **if** ($0 \in F(X)$) — *Traditional interval rejection rule*
6. **if** ($w(X) \leq \epsilon$)
7. $Q := Q + \{X\}$ — *X belongs to the final solution*
8. **if** (GSC) $L := \{\}$ — *GSC (Global Sign Change) test*
9. **else**
10. Bisect(X, X_l, X_r) — $X_l = [\underline{x}, m(X)]$, $X_r = [m(X), \overline{x}]$
11. $L := L + \{X_r\} + \{X_l\}$ — *Subinterval are stored in Head(L)*
12. **if** ($\overline{[F(\underline{x_l}) \cdot F(\overline{x_l})]} \leq 0$)) — *SC (Sign Change) test*
13. Remove all $X_i \in L : \underline{x_i} \geq \overline{x_l}$
14. **return** Q

$[\underline{x}, \overline{x}, \underline{F}(X), \overline{F}(X)]$. Three types of boxes can be distinguished among the evaluated intervals. Light grey boxes are those which were rejected by the traditional rejection rule, white ones represent the set of subdivided intervals and dark grey boxes are the set of intervals in the final list Q which defines the solution.

Algorithm 2 (Minimal Root Finder (MRF)) is focused on finding not all the roots of the function but only the minimal one. It is carried out by removing from the work list L all the intervals that are known not to contain the minimal root. In addition to the traditional rejection rule $(0 \notin F(X))$, Algorithm 2 incorporates the following rejection tests:

SC (Sign Change) test: An interval $X : \overline{[F(\underline{x}) \cdot F(\overline{x})]} \leq 0$ contains at least one root, so all intervals at the right of $\overline{x}$ can be rejected (Algorithm 2, line 12).

GSC (Global Sign Change) test: When an interval is stored in the final list and it belongs to a continuous set of final intervals, the SC test is applied to interval defined by the whole continuous set. Notice that single intervals X, that do not satisfy the SC test can belong to a set of continuous intervals which verifies the GSC test. Figure 2.2 shows an example of a set of two

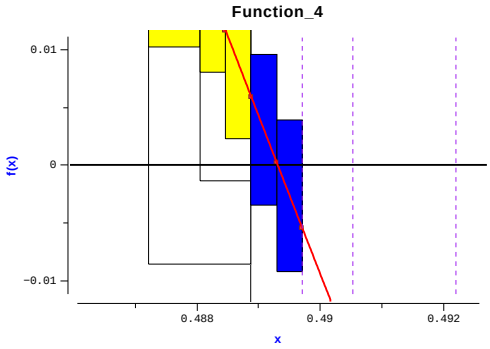


FIG. 2.2. *Zoom of the graphical output generated by Algorithm 3 for the problem number 4 using* $\epsilon = 10^{-4} \cdot w(X)$

intervals fulfilling GSC test.

For differentiable functions, the convergence could be accelerated by taking advantage of the quadratic convergence of the interval Newton method [9, 19, 20], or applying the following test:

EqualSignMonotonic test: When $0 \notin F'(X)$ and $\underline{[F(\underline{x}) \cdot F(\overline{x})]} > 0$, the interval X can be rejected because $0 \notin f(X)$.

Algorithm 2 follows the traditional order of calculations proposed by Mitchell without derivative evaluations: First, evaluating the inclusion function at interval X $(F(X))$; second, the value of the function at $\overline{x}$ $(F(\overline{x}))$; notice that $F(\underline{x_l})$ is only evaluated for the first interval $X_l \subset S : \underline{x_l} = \underline{s}$ because for the other intervals $X \subset S$, $F(\underline{x})$ has previously been evaluated. Nevertheless for the problem of finding the minimal root of a function, this is not necessarily the most appropriate order. In this work we propose to use the opposite order based on the assumption that intervals X verifying that $0 \in [F(\underline{x}) \cdot F(\overline{x})]$ also verify that $0 \in F(X)$, so it is not necessary to evaluate $F(X)$.

The new version of the minimal root finder algorithm with the reordered interval evaluations (called MRFro) is described by Algorithm 3. In Algorithm 2, $F(X)$ is evaluated (line 5) before carrying out the interval point evaluation $(F(\underline{x}) \cdot F(\overline{x}))$ (line 12). For a selected interval X, $F(X)$ is always evaluated. In Algorithm 3, an interval point evaluation is done in line 6, before evaluating $F(X)$ in line 9. Now, if the condition in line 6 is satisfied, $F(X)$ is not evaluated.

As an example, in Figure 2.3, graphical representations of the execution of RF, MRF and MRFro algorithms are shown. A dot on the profile of the function at x means that $F([x, x])$ has been evaluated. Vertical dashed lines highlight when algorithms (MRF and MRFro) take advantage of the information provided by $F([x_i, x_i])$ to eliminate those intervals $X : \underline{x} \geq x_i$.

3. Numerical comparison of the methods working with a single function. In this Section, in order to illustrate the performance of MRF and MRFro in comparison with RF, a series of numerical experiments is presented. Evaluations of the computational cost of RF, MRF and MRFro were made by using a set of forty test functions, whose description is given in Table 3.1, where the following notations for column headers is used:

N : Index of the function (for further references).

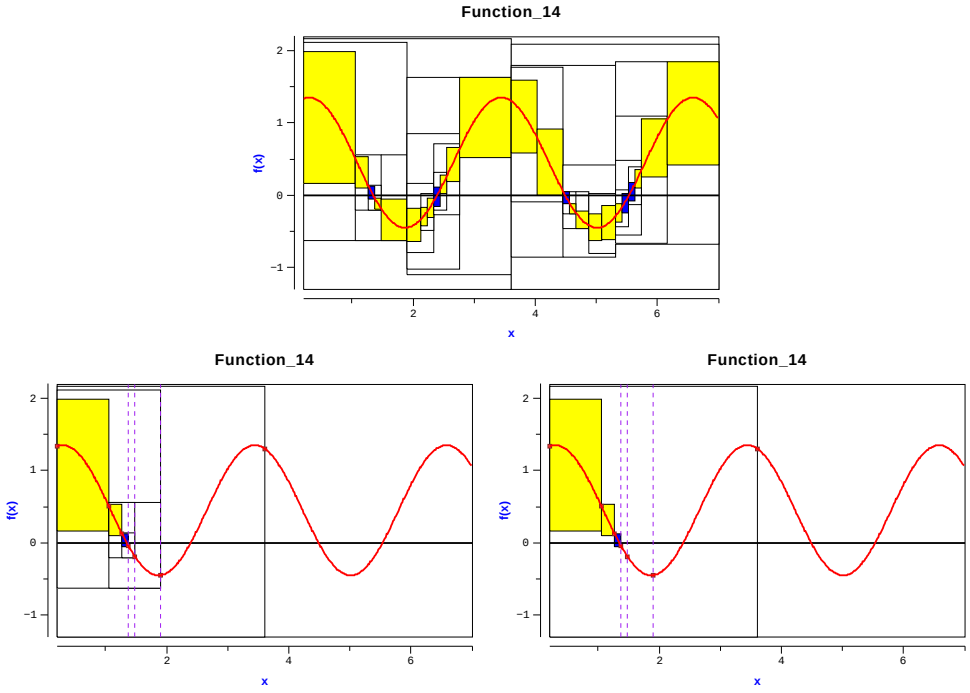


FIG. 2.3. *Graphical results from the execution of several versions of root finder and the minimal root finder algorithms ($\epsilon = 0.02 \cdot w(X)$). From top to bottom and left to right: algorithms RF, MRF and MRFro.*

MR : Location of the minimal root of the function rounded to four decimal digits.
NR : Number of roots of f in S.
NM : Number of local extrema (max. and min.).

It can be seen in Table 3.1 that all the test functions are multiextremal, most of them contain several roots, and thirteen of the functions have one or several points of non-differentiability.

Table 3.2 shows numerical results obtained from executions of the previously described algorithms applied to the set of functions in Table 3.1. Columns with header IE refer to the number of interval function evaluations carried out by the corresponding algorithm, as an indicator of the computational effort applied to solve (1.1). Columns NS specify the number of isolated solutions found by the algorithms using the format $x+y$, where x is the number of solutions without a verified zero, and y is the number of solutions with a verified zero. Columns NQ provide information on the number of intervals in the final list. Notice that when $NS = 0 + 1$ or $NS = 1 + 0$, the accuracy of the solution is actually $\epsilon \cdot NQ$. Columns with headers RF, MRF, and MRFro provide numerical results obtained by Algorithms 1, 2, and 3, respectively. Figures at the bottom row show the average values of the number of interval evaluations.

For functions without zero crossing points, all the algorithms compute almost the same number of interval evaluations. An exception happens for function N=33, where RF is better than the remaining algorithms. This is because MRF and MRFro algorithms do not eliminate any interval by the SC or GSC tests.

Algorithm 3 : The Minimal Root Finder reordered Algorithm

Funct MRFro (S, F, ϵ, L, Q)

1. $Q := \{\}$; $L := \{S\}$ — *Final and Work Lists*
2. **while** ($L \neq \{\}$)
3. $X :=$ Head(L)
4. $L := L - \{X\}$
5. Rejected := True — *The current interval could be rejected*
6. **if** ($\overline{[F(\underline{x}) \cdot F(\overline{x})]} \leq 0$) — *SC (Sign Change) test*
7. Remove all $X_i \in L : \underline{x_i} \geq \overline{x}$
8. Rejected := False
9. **elsif** ($0 \in F(X)$) — *Traditional interval rejection rule*
10. Rejected := False
11. **if** (Rejected = False)
12. **if** ($w(X) \leq \epsilon$)
13. $Q := Q + \{X\}$ — *X belongs to the final solution*
14. **if** (GSC) $L := \{\}$ — *GSC (Global Sign Change) test*
15. **else**
16. Bisect (X, X_l, X_r) — $X_l = [\underline{x}, m(X)]$, $X_r = [m(X), \overline{x}]$
17. $L := L + \{X_r\} + \{X_l\}$ — *Subintervals are stored in Head(L)*
18. **return** Q

For three out of four of the functions with only one root $(8, 20, 22)$, RF outperforms MRF but is worse than MRFro. Of course, for functions with several roots RF works significantly worse than the rest of the algorithms because it was designed to find all the roots of the function. It can be seen in Table 3.1 that MRFro is computationally less expensive than MRF for all the functions with at least root in the search space. Nevertheless, no difference can be found in the values of NS or NQ obtained by MRF and MRFro algorithms. For functions without roots in the search space all the algorithms are computationally very inexpensive.

In order to use the EqualSignMonotonic test, derivative evaluations were included in the MRFro algorithm (MRFro+). In addition, uniqueness can be verified using derivative information for those intervals at the final list Q such that $\overline{F(\underline{x}) \cdot F(\overline{x})} \leq 0$ AND $0 \notin F'(X)$. Notice that MRFro+ evaluates $F'(X)$ and $F(X)$ only when $\overline{F(\underline{x}) \cdot F(\overline{x})} > 0$. The version of Mitchell's algorithm implemented does not include conventional numerical methods for the root refinement. However, it includes the EqualSignMonotonic test by evaluating $F(\underline{x}) \cdot F(\overline{x})$ only when $0 \notin F'(X)$ or $X \in Q$. Hence, the only difference between both algorithms consists of the order of computations carried out on any interval. For Mitchell's algorithm the ordering is $F(X), F'(X)$ and $F(\underline{x}) \cdot F(\overline{x})$, while for the MRFro+ algorithm is the opposite: $F(\underline{x}) \cdot F(\overline{x})$, $F'(X)$ and $F(X)$. The SC and GSC tests have also been included in Mitchell's algorithm because the stopping criterion to find the minimal root was not specified in [17]. For both algorithms, when $0 \notin F'(X)$, $F'(Y) : Y \subset X$ is not evaluated because $0 \notin F'(Y)$ is assured.

In Table 3.2 numerical results obtained by MRFro+ and Mitchell's algorithm are shown. Executions were carried out only for the subset of functions in Table 3.1 which do not have any point of non differentiability. It can be seen that for several functions, the number of interval function evaluations carried out by the MRFro algorithm (without monotonicity test) is lower than those computed by Mitchell's

TABLE 3.1

Description of the test functions. Functions marked with ▷ have points of non-differentiability over $S = [0.2, 7]$. They are ordered by their minimal root. In Appendix A functions are graphically shown.

	N	Function $f(x)$	MR	NR	NM
	1	$-e^{\sin(3x)} + 2$	0.2553	7	9
	2	$\ln(3x)\ln(2x) - 0.1$	0.2804	2	3
▷	3	$\begin{cases} \cos(5x) & x \le 3\pi/2 \\ \cos(x) & otherwise \end{cases}$	0.3142	8	10
	4	$-\sum_{k=1}^{5} k\sin[(k+1)x + k] + 3$	0.4893	10	15
	5	$e^{-x}\sin(2\pi x)$	0.5000	14	15
	6	$-\sum_{k=1}^{10} 1/(\lvert k_i(x - a_i)\rvert + c_i) + 10$	0.6426	4	15
▷		a=(3.040,1.098,0.674,3.537,6.173,8.679,4.503,3.328,6.937,0.700)			
		k=(2.983,2.378,2.439,1.168,2.406,1.236,2.868,1.378,2.348,2.268)			
		c=(0.192,0.140,0.127,0.132,0.125,0.189,0.187,0.171,0.188,0.176)			
	7	$x + \sin(5x)$	0.8209	2	13
	8	$-x + \sin(3x) + 1$	1.0354	1	9
▷	9	$\begin{cases} \sin(3x) & x \le 5 \\ 7x^2 - 14x - 100 & otherwise \end{cases}$	1.0472	4	8
	10	$\cos(x) + 2\cos(2x)e^{-x}$	1.1407	2	4
	11	$-\sqrt{x}\sin(x) + 1$	1.1748	3	4
	12	$(x^2 - 5x + 6)/(x^2 + 1) - 0.5$	1.2583	1	3
	13	$(3x - 1.4)\sin(18x) + 1.7$	1.2655	34	42
	14	$\sin(x)\cos(x) - 1.5\sin^2(x) + 1.2$	1.3408	4	7
	15	$(x+1)^3/x^2 - 7.1$	1.3646	2	3
▷	16	$-\lvert x\sin(x)\rvert + 1.5$	1.5034	5	6
▷	17	$\begin{cases} x^2 + 0.5 & x \le 1 \\ 5\sin(2\pi x) + 1.5 & 1 < x \le 3 \\ x^2 - 8x + 16.5 & otherwise \end{cases}$	1.5485	4	8
	18	$\cos(x) - \sin(5x) + 1$	1.5708	6	13
▷	19	$\begin{cases} \cos(5x) & 3\pi/2 \le x \le 5\pi/2 \\ \cos(x) & otherwise \end{cases}$	1.5708	5	7
	20	$-x - \sin(3x) + 1.6$	1.9686	1	9
	21	$x\sin(x) + \sin(10x/3) + \ln(x) - 0.84x + 1.3$	2.9609	2	6
	22	$-0.5x^2\ln(x) + 5$	3.0117	1	2
▷	23	$x\lvert\sin(x)\rvert - 0.02$	3.1352	4	6
	24	$2\sin(x)e^{-x}$	3.1416	2	4
▷	25	$\begin{cases} \sin(x) & x \le \pi \\ \sin(5x) & otherwise \end{cases}$	3.1416	7	9
	26	$\sqrt{x}\sin^2(x)$	3.1416	2	6
▷	27	$\begin{cases} \sin(x) & \sin(x) > \cos(x) \\ \cos(x) & otherwise \end{cases}$	3.1416	2	6
	28	$2(x-3)^2 - e^{x/2} + 5$	3.2811	2	4
	29	$\begin{cases} \sin(5x) + 2 & x \le \pi \\ 5\sin(x) + 2 & x > \pi \end{cases}$	3.5531	2	8
	30	$\sum_{k=0}^{5} k\cos[(k+1)x + k] + 12$	4.7831	2	15
	31	$\sum_{k=1}^{5} -\cos[(k+1)x] + 4$	6.1304	2	14
▷	32	$\lvert\sin^3(x)\cos^3(x)\rvert + 0.1$	-	-	10
▷	33	$\begin{cases} -x + 4 & x \le 3 \\ -8/9x^2 + 8x - 15 & 3 < x < 6 \\ x - 5 & otherwise \end{cases}$	-	-	5
▷	34	$\lvert\sin(x) + \cos(x)\rvert + 0.3$	-	-	6
▷	35	$x + \lvert\sin(x)\cos(x)\rvert$	-	-	10
	36	$-e^{\sin(x)} + 4$	-	-	4
	37	$e^{\sin(3x)}$	-	-	9
	38	$2\cos(x) + \cos(2x) + 5$	-	-	6
	39	$-x\sin(x) + 5$	-	-	4
	40	$-e^{-x}\sin(2\pi x) + 1$	-	-	13

TABLE 3.2
Numerical results for algorithms. $\epsilon = w(S) \cdot 10^{-15}$.

	RF			MRF			MRFro		
N	IE	NS	NQ	IE	NS	NQ	IE	NS	NQ
1	675	7+0	10	124	0+1	1	77	0+1	1
2	191	2+0	2	123	0+1	1	77	0+1	1
3	763	8+0	14	122	0+1	1	74	0+1	1
4	2207	10+0	39	214	0+1	2	163	0+1	2
5	1333	14+0	19	125	0+1	1	76	0+1	1
6	883	8+0	11	122	0+1	1	78	0+1	1
7	287	2+0	4	192	0+1	1	146	0+1	1
8	105	1+0	1	125	0+1	1	75	0+1	1
9	397	4+0	5	132	0+1	1	82	0+1	1
10	255	2+0	4	189	0+1	2	134	0+1	2
11	349	3+0	5	130	0+1	1	81	0+1	1
12	241	1+0	2	194	0+1	2	144	0+1	2
13	3137	34+0	54	135	0+1	1	91	0+1	1
14	519	4+0	8	128	0+1	1	79	0+1	1
15	1441	2+0	18	616	0+1	6	564	0+1	6
16	575	5+0	8	123	0+1	1	73	0+1	1
17	451	4+0	5	125	0+1	2	75	0+1	2
18	803	6+0	12	127	0+1	2	81	0+1	2
19	487	5+0	11	127	0+1	2	76	0+1	2
20	133	1+0	1	173	0+1	1	123	0+1	1
21	491	2+0	9	160	0+1	2	106	0+1	2
22	101	1+0	2	127	0+1	2	76	0+1	2
23	369	4+0	7	121	0+1	1	80	0+1	1
24	201	2+0	3	126	0+1	1	76	0+1	1
25	677	7+0	13	133	0+1	2	79	0+1	2
26	201	2+0	3	302	2+0	3	298	2+0	3
27	199	2+0	3	126	0+1	1	76	0+1	1
28	1051	2+0	13	205	0+1	2	155	0+1	2
29	199	2+0	2	127	0+1	1	77	0+1	1
30	265	2+0	5	204	0+1	3	154	0+1	3
31	209	2+0	4	143	0+1	2	93	0+1	2
32	1	0+0	0	2	0+0	0	2	0+0	0
33	39	0+0	0	59	0+0	0	59	0+0	0
34	1	0+0	0	2	0+0	0	2	0+0	0
35	1	0+0	0	2	0+0	0	2	0+0	0
36	1	0+0	0	2	0+0	0	2	0+0	0
37	1	0+0	0	2	0+0	0	2	0+0	0
38	1	0+0	0	2	0+0	0	2	0+0	0
39	3	0+0	0	5	0+0	0	5	0+0	0
40	1	0+0	0	2	0+0	0	2	0+0	0
M.	**481**			**130**			**93**		

	MRFro+			Mitchell		
N	IE	NS	NQ	IE	NS	NQ
1	80	0+1	1	126	0+1	1
2	83	0+1	1	125	0+1	1
3						
4	92	0+1	2	131	0+1	2
5	77	0+1	1	127	0+1	1
6						
7	100	0+1	1	131	0+1	1
8	75	0+1	1	127	0+1	1
9						
10	92	0+1	2	139	0+1	2
11	82	0+1	1	132	0+1	1
12	98	0+1	1	138	0+1	1
13	103	0+1	1	137	0+1	1
14	80	0+1	1	127	0+1	1
15	141	0+1	2	162	0+1	2
16						
17						
18	88	0+1	2	129	0+1	2
19						
20	112	0+1	1	163	0+1	1
21	107	0+1	2	153	0+1	2
22	76	0+1	2	129	0+1	2
23						
24	78	0+1	1	128	0+1	1
25						
26	398	2+0	3	310	2+0	3
27						
28	103	0+1	1	138	0+1	1
29						
30	124	0+1	2	154	0+1	2
31	102	0+1	2	145	0+1	2
32						
33						
34						
35						
36	3	0+0	0	1	0+0	0
37	3	0+0	0	1	0+0	0
38	3	0+0	0	1	0+0	0
39	7	0+0	0	4	0+0	0
40	3	0+0	0	1	0+0	0
M.	**89**			**118**		

algorithm, but for function N=15, MRFro is almost five times more expensive than Mitchell's and MRFro+ algorithms.

In many cases MRFro+ is computationally less expensive than Mitchell's algorithm. Only for function $N = 26$, Mitchell's algorithm clearly outperforms MRFro+. This is due to the fact that the function $N = 26$ is non negative but has two roots in the search region. The rest of the functions where Mitchell's algorithm outperforms MRFro+ do not have any roots, and so intervals were rejected only by the traditional rejection rule. In this case MRFro+ carries out more evaluations of the inclusion function because the function is evaluated at the bounds of the interval before computing the value of $F(X)$.

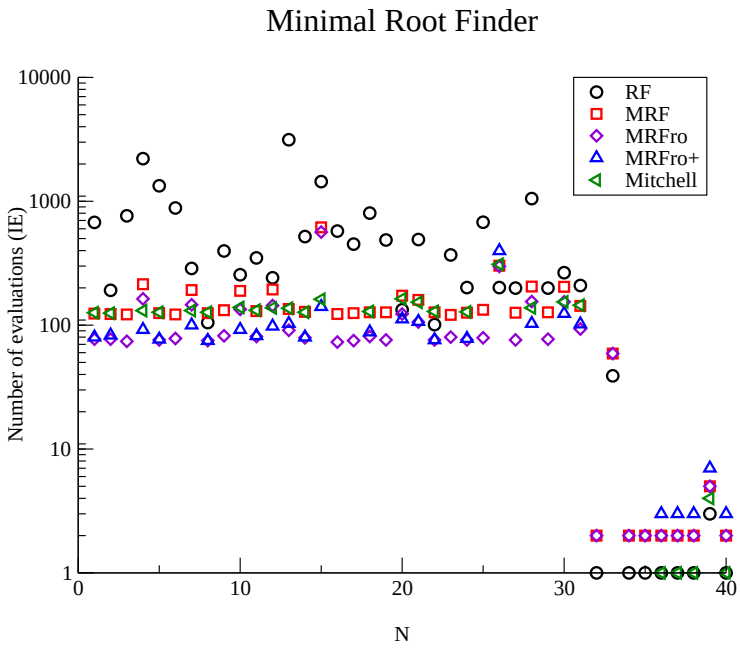


FIG. 3.1. *Graphic representation of the data contained in Table 3.2.*

Summing up, numerical results have shown that on average MRFro is almost 30% faster than MRF, and MRFro+ is 24% faster than Mitchell's algorithm. It means that the sequence of evaluations $(F(\underline{x}) \cdot F(\overline{x}))$, $F'(X)$ and $F(X)$, produces a faster convergence to the solution. From our point of view, these results are related to the fact that the overestimations produced by Interval Arithmetic usually decrease with the width of the interval; i.e. the most precise information is obtained from intervals $[x, x]$. Therefore, it is better to make decisions based on the evaluation of $F([x, x])$ than on the evaluation of $F(X)$.

4. Extending the MRFro algorithm for finding the minimal root of several functions. This section is devoted to analyzing several algorithmic solutions to the problem (1.3). The simplest and computationally most expensive way to solve such a problem consists of decomposing the problem into p independent (1.1) problems, solving them by the above described algorithms and keeping the minimum value of the set $\{t_i^*,\ i = 1, \ldots, p\}$ of individual solutions.

However, there exist several ways to improve this coarse approach. In this section, an extension of the MRFro algorithm (MRFroEx) that solves problem (1.3) is analyzed. The main idea in our algorithm consists of using the information obtained from any of the functions to determine the set of intervals where a solution cannot be found; thus reducing the search domain associated to all the functions. Our proposal for solving (1.3) is described by Algorithm 4, where each function $F_i,\ i = 1, \ldots, p$, has associated its own work list (L_i) and final list (Q_i). The algorithm iterates sequentially until all the work lists L_i become empty.

At every iteration, the algorithm will work on an interval X, selected from the subset of intervals which verify that $\underline{x} = \min\{\underline{x_j},\ \forall X_j \in L_i,\ \forall\ i = 1, \ldots, p\}$, ensuring that the search is focused on finding the minimal root. If the subset contains several

Algorithm 4 : MRFro algorithm for a set of p functions (MRFroEx)

Funct MRFroEx($S = [\underline{s}, \overline{s}], F_1, \ldots, F_p, \epsilon, L_1, \ldots, L_p, Q_1, \ldots, Q_p$)

1. $Q_i := \{\}$; $L_i := \{S\}$; $i = 1, \ldots, p$ *Initializing Work and Final Lists*
2. $s_{min} := \underline{s}$; $s_{max} := \overline{s}$ *Active search region*
3. **while** ($\exists L_j \neq \{\} : j = 1, \ldots, p$)
4. **for** ($L_j : L_j \neq \{\}$ **and** $s_{min} = \underline{y} : Y = Head(L_j)$)
5. $X :=$ Head(L_j)
6. $L_j := L_j - \{X\}$
7. Rejected := False *The current interval could be rejected*
8. **if** (**not** Eval($F_j(\overline{x})$) **or** (Eval($F_j(\overline{x})$) **and** $\underline{F_j(\underline{x}) \cdot F_j(\overline{x})} > 0$))
9. **if** ($0 \notin F_j(X)$)
10. UpdateSmin($s_{min}, L_1, \ldots, L_p, \overline{s}$)
11. Rejected := True
12. **if** (Rejected = False)
13. **if** ($w(X) \leq \epsilon$)
14. UpdateSmin($s_{min}, L_1, \ldots, L_p, \overline{s}$)
15. $Q_j := Q_j + \{X\}$
16. **if** (GSC) *GSC (Global Sign Change) test*
17. $s_{max} = \overline{x}$
18. UpdateLists($s_{max}, L_1, \ldots, L_p$)
19. **else**
20. Improve := UpdateSmax(s_{max}, X)
21. Bisect(X, X_l, X_r)
22. $L_j := L_j + \{X_r\} + \{X_l\}$
23. **if** (Improve)
24. UpdateLists($s_{max}, L_1, \ldots, L_p$) *SC (Sign Change) test*
25. **return** $Q := Q_1, \ldots, Q_p$

Algorithm 5 : Check the update of the value of s_{min}

Proc UpdateSmin($s_{min}, L_1, \ldots, L_p, \overline{s}$)

1. $s_{min} := \overline{s}$
2. **for** ($i := 0 \ldots p$)
3. **if** ($L_i \neq \{\}$)
4. $X :=$ Head(L_i)
5. **if** ($\underline{x} < s_{min}$)
6. $s_{min} := \underline{x}$

intervals, the selection of the objective interval is made by an a priory established order.

The active search region is defined by the interval $[s_{min}, s_{max}]$ where the solution of (1.3), if any, can be found. Therefore, at the beginning $s_{min}=\underline{s}$ and $s_{max}=\overline{s}$ (line 2) and during the execution of the algorithm $s_{min} = \min\{\underline{x_j},\ \forall X_j \in L_i,\ \forall\ i = 1, \ldots, p\}$ and $s_{max} = \max\{\overline{x_j},\ \forall X_j \in L_i,\ \forall\ i = 1, \ldots, p\}$. The active interval, $[s_{min}, s_{max}]$, is updated at every iteration (see Algorithms 5 and 6).

The algorithm takes advantage of the information obtained from one of the functions to reduce the search domain of the remaining functions. If a function f_i verifies the SC test or GSC test in an interval X, it can be assured that the minimal root is located at a point $y : y \leq \overline{x}$, so the active region can be updated ($s_{max} = \overline{x}$), and

Algorithm 6 : Check the update of the value of s_{max}

Funct UpdateSmax(s_{max}, X)
1. Improve := False
2. **if** ($\overline{F(\underline{x}) \cdot F(m(X))} \leq 0$)
3. $s_{max} := m(X)$
4. Improve := True
5. **return** Improve

Algorithm 7 : Update the work lists based on s_{max}

Proc UpdateLists($s_{max}, L_1, \ldots, L_p$)
1. **for** ($i := 0 \ldots p$)
2. **if** ($L_i \neq \{\}$)
3. Remove $X_j \in L_i : \underline{x_j} \geq s_{max}$
4. **if** ($L_i \neq \{\}$) *There is only one interval in the list*
5. $X := \text{Head}(L_i)$
6. $X := [\underline{x}, s_{max}]$

Algorithm 8 : Check the update of the value of s_{max} and repeat the process on $[\underline{x}, m(X)]$ if $\overline{F(\underline{x}) \cdot F(m(X))} \leq 0$)

Funct UpdateSmax2(s_{max}, X)
1. Improve := False
2. **while** ($w(X) > \epsilon$ **and** $(\overline{F(\underline{x}) \cdot F(m(X))} \leq 0)$)
3. $s_{max} := m(X)$
4. $X := [\underline{x}, s_{max}]$
5. Improve := True
6. **return** Improve

the search space on the right of $\overline{x}$ can be eliminated from all the working lists L_i. This is made by removing the intervals $X_j : \underline{x_j} \geq s_{max}$ and reducing the intervals $X_j : s_{max} \in X_j$, to $[\underline{x_j}, s_{max}]$, $\forall X_j \in L_i$, $\forall L_i$.

On the other hand, the value of s_{min} will only be changed if the traditional elimination rule is able to discard the full subset of intervals X verifying that $\underline{x} = s_{min}$.

MRFroEx algorithm computes two kinds of interval function evaluations: $F(X)$ and $F(m(X))$; when working on function F_j, $F_j(X)$ will only be evaluated if the value of $F_j(\overline{x})$ is not known (**not** Eval($F_j(\overline{x})$), line 8 in Algorithm MRFroEx) or $F_j(\overline{x})$ is known and X does not satisfy the SC test. The situations where the value of $F_j(\overline{x})$ is not known appear when $X = S$ or a function f_i, $i \neq j$, previously examined, verified that $\overline{F_i(\underline{x}) \cdot F_i(\overline{x})} \leq 0$. In both cases $F_j(\overline{x})$ is not evaluated because the information provided does not allow reduction of the search space. If X was not eliminated from the work list L_j, $F_j(m(X))$ is evaluated if $w(X) > \epsilon$.

Algorithm MRFroEx has been designed as an extension of MRFro for working on a set of p functions. In a similar way, an extended version of MRF (MRFEx) for working on a set of p functions has been implemented.

A possible improvement of the Algorithm 4 consists of repeatedly updating the value of s_{max} using the same interval from the same function until s_{max} cannot be improved, instead of moving computations to the following available function. By incorporating this idea in the algorithm MRFroEx, the new algorithm MRFroEx-II

TABLE 5.1

Number of interval function evaluations for algorithms MRFEx (column A), MRFroEx (column B) and MRFroEx-II (Column C). Accuracy was $\epsilon = w(S)\cdot 10^{-15}$ for all the cases. Numerical results were obtained for three different orders of the set of functions. Row at location i contains numerical results of the new algorithms using a subset of i functions; the subset of functions specifies in column N from row 1 to row i.

Ordered form 1 to 40			
N	**A**	**B**	**C**
1	124	77	77
2	137	92	88
3	147	101	97
4	149	104	106
5	155	109	104
6	161	115	108
7	167	121	112
8	157	112	107
9	161	116	111
10	165	120	115
11	167	122	117
12	171	126	121
13	175	130	125
14	177	132	127
15	181	136	131
16	183	138	133
17	185	140	135
18	187	142	137
19	189	144	139
20	193	148	143
21	197	152	147
22	199	154	149
23	201	156	151
24	203	158	153
25	205	160	155
26	207	162	157
27	209	164	159
28	211	166	161
29	213	168	163
30	217	172	167
31	219	174	169
32	221	176	171
33	223	178	173
34	225	180	175
35	227	182	177
36	229	184	179
37	231	186	181
38	233	188	183
39	235	190	185
40	237	192	187

Ordered from 40 to 1			
N	**A**	**B**	**C**
40	2	2	2
39	7	7	7
38	9	9	9
37	11	11	11
36	13	13	13
35	15	15	15
34	17	17	17
33	76	76	76
32	78	78	78
31	218	168	168
30	279	229	229
29	196	146	146
28	284	234	234
27	206	156	156
26	328	278	231
25	380	354	306
24	447	421	381
23	283	236	239
22	270	220	221
21	302	249	250
20	308	258	258
19	270	220	221
18	348	319	298
17	290	242	245
16	292	245	248
15	784	732	733
14	347	299	300
13	356	309	307
12	414	366	361
11	350	303	301
10	404	353	356
9	331	283	285
8	343	295	297
7	381	333	335
6	303	256	256
5	256	209	210
4	353	304	308
3	251	205	209
2	251	206	211
1	257	213	219

Randomly ordered			
N	**A**	**B**	**C**
19	127	76	76
4	220	168	166
6	226	174	170
39	228	176	172
24	232	180	174
12	236	185	178
2	153	106	96
29	155	108	98
37	157	110	100
11	161	114	104
26	163	116	106
34	165	118	108
27	167	120	110
17	171	124	114
21	177	130	118
28	179	132	120
22	181	134	122
5	187	140	126
15	193	146	130
36	195	148	132
9	183	137	137
7	187	141	141
8	191	145	145
40	193	147	147
31	195	149	149
32	197	151	151
1	205	160	155
30	207	162	157
23	209	164	159
33	211	166	161
35	213	168	163
3	213	169	165
18	215	171	167
25	217	173	169
16	219	175	171
13	221	177	173
20	223	179	175
14	225	181	177
38	227	183	179
10	229	185	181

was obtained. This is simply made by using Algorithm 8 instead of Algorithm 6 (Algorithm 4, line 20).

5. Numerical experiments with the algorithms finding the minimal root of several functions. Implementations of the methods[1] MRFEx, MRFroEx, and MRFroEx-II have been carried out and tested by using the set of functions described in Table 3.1. Performance evaluations for MRFEx, MRFroEx, and MRFroEx-II are shown in Table 5.1, and a graphical representation of the same experimental results is shown in Figure 5.1.

Table 5.1 and Figure 5.1 contain the results of three series of experiments. All

[1] The code is available from the authors.

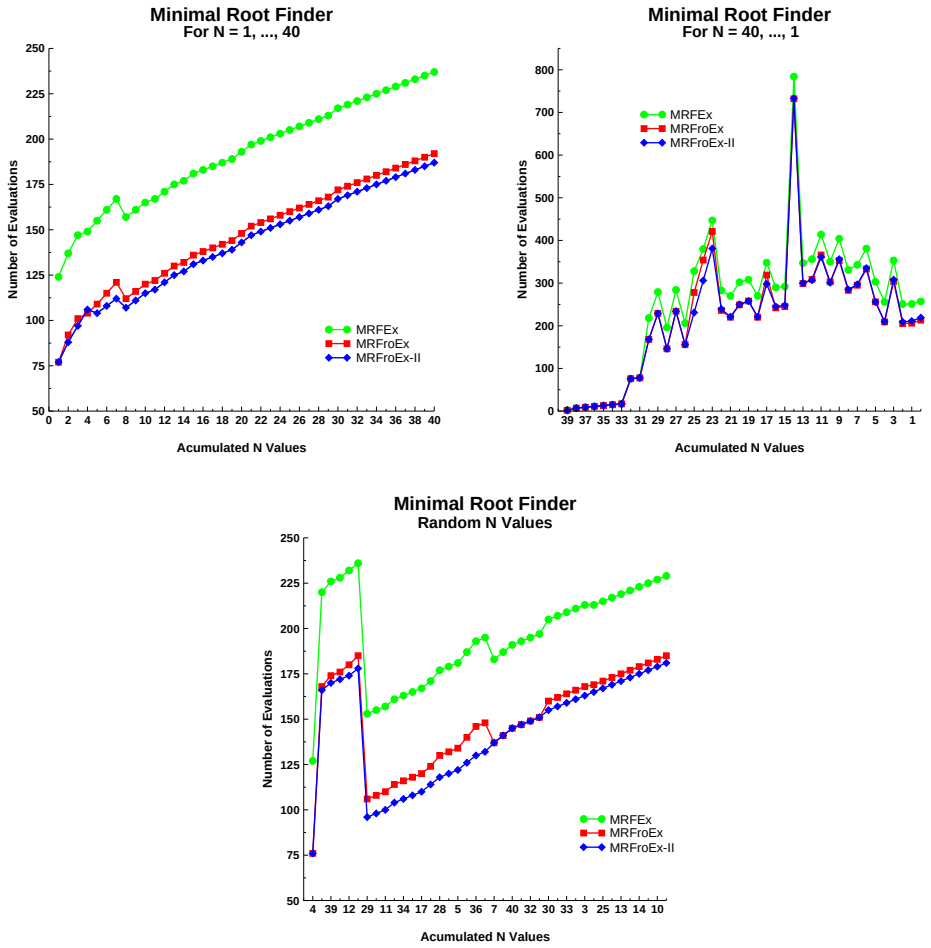


FIG. 5.1. *Graphic representation of the data contained in Table 5.1.*

the algorithms were tested for forty different values of p (number of functions in the set, $p = 1, 2, 3, \ldots, 40$).

- In the first series, the set containing p test functions includes functions $N = 1, \ldots, p$. For example, data in the third row ($N = 3$), means that the set of tested functions were 1, 2, and 3. These data are shown in the first subtable of Table 5.1 and at the upper left hand graph of Figure 5.1. Notice that functions are ordered by the location of their minimal root. When p increases, a slowly but almost linear increase of the number of function evaluations is observed. This is mainly due to the improvements in the value of s_{max}, which after a few iterations is small enough and the resulting active interval does not contain the minimal roots for most of the functions. These functions will be inactive in a few iterations due to the traditional elimination rule.
- In the second series, the sets of functions were built in a decreasing order based on the values of N, always starting with $N = 40$. For example, the set containing three functions (3^{rd} row) includes functions $N = 40, 39, 38$, and so on. These data are shown in the second subtable of Table 5.1 and at the

upper right hand graph of Figure 5.1. In this series of experiments, the last function in the set contains the solution of the problem (1.3). Therefore, at the end of the search, algorithms usually work only on that function alone. Similarly to the previous case, the number of evaluations made on the function with the minimal root will be an important factor in the final result. Two significant particular cases can be distinguished in the data: when the last function is number 15 and when the last function is number 24. In the first case, the number of interval evaluations is significantly increased because the overestimations of the real range obtained by the inclusion function are very large. Only when the intervals are very small can they be rejected by the traditional elimination rule. The number of evaluations can be reduced if monotonicity test is used, as was shown in Table 3.2. In the second case, functions 27, 26, 25, and 24 have the same value of their own minimal root in the search region. Therefore, the algorithms have to work on all of them all the time, with a corresponding increase in the number of interval evaluations.

- In the third series of experiments, the set of functions were randomly ordered, and a set with p functions were built by taking the p first functions. These data are shown in the third subtable of Table 5.1 and at the bottom graph of Figure 5.1. These experiments have been introduced because, usually, the order of the functions, based on location of the minimal root, is not a priory known. As in the previous cases, in these experiments, the ordering of the functions and the function having the minimal root in the set, are very important for improving the value of s_{max} and for reducing the number of interval function evaluations.

By comparing the three algorithms, it can be observed that MRFroEx-II and MRFroEx versions are always faster than the MRFEx one, as could be expected from the results shown in Section 3. Only small differences between MRFroEx and MRFroEx-II can be noticed.

Numerical results show that the strategy investigated here to solve problem (1.3) exhibits a fast convergence.

6. Conclusion. Two problems very often arising in several scientific and engineering applications have been considered in this paper. The first problem consists of finding the minimal root of an analytic one-dimensional function over a given interval. It is supposed that the objective function can be multiextremal and non-differentiable.

Two new algorithms have been proposed to solve this problem. The novelty of the new methods lies in improving the elimination criteria and the order in which interval and point evaluations are executed. The first algorithm (MRF) concentrates its attention on finding the first root instead of looking for all the roots as traditional methods do. The second method (MRFro) adds an additional acceleration to the search by introducing the following idea.

Traditional algorithms for finding roots of equations evaluate first $F(X)$, then $F'(X)$, and, finally, $F([x,x])$ in a point x. In this paper, it has been shown that it is better to carry out $F([x,x])$ evaluations first (mainly when there is at least one root in the search region). This new order of evaluation accelerates the search because the evaluation of the inclusion function obtained by interval arithmetic usually produces overestimations of the real range of the function, due to the definition of operations in the Interval Arithmetic and also to the outward rounding used by Computational Interval Arithmetic. Usually this overestimation decreases linearly with the width of the interval. Thus, the most precise information can be obtained from intervals $[x,x]$.

Therefore, when the length of the interval X is greater than zero, it is better to start evaluating $F([x,x])$ than $F(X)$.

For problems with multiextremal and differentiable objective functions, an extension of the method MRFro – the algorithm MRFro+ – has been proposed. Numerical experiments carried out on a wide set of test functions demonstrate a quite satisfactory performance of the three new algorithms in comparison with methods taken from literature.

The second problem considered in the paper is a generalization of the first one and deals with the search of the minimal root in a set of N multiextremal and non-differentiable functions. The methods proposed for solving the first problem have been generalized for solving the second. The main idea of the generalization is in simultaneous work with all N functions on the same search domain. The search information obtained from any of the functions is immediately applied to all the functions and allows us to reduce the search domain for all of them. Numerical experiments have shown that this idea significantly accelerates the search.

Acknoledgements. The authors would like to thank the referees for their useful remarks and suggestions.

REFERENCES

[1] Baoxin Li, A. Law, H. Raafat, P. Nguyen, and Y.-F. Yan, *Eigenvalues of tridiagonal matrices: An alternative to Givens' methods*, Computers Math. Applic., 19 (1990), pp. 89–94.

[2] D. Bedrosian and J. Vlach, *Time-domain analysis of networks with internally controlled switches*, IEEE Transactions on Circuits Syst., CAS-39 (1992), pp. 199–212.

[3] O. Caprani, L. Hvidegaard, M. Mortensen, and T. Schneider, *Robust and efficient ray intersection of implicit surfaces*, Reliable Computing, 1 (2000), pp. 9–21.

[4] L. G. Casado, I. García, and Y. D. Sergeyev, *Interval branch and bound algorithm for finding the First-Zero-Crossing-Point in one-dimensional functions*, Reliable Computing, 2 (2000), pp. 179–191.

[5] J. Clark, *Authenticating edges produced by zero-crossing algorithms*, IEEE Transactions on Pattern Analysis and Machine Intelligence, PAMI, 11 (1989), pp. 43–57.

[6] P. Daponte, D. Grimaldi, A. Molinaro, and Y. D. Sergeyev, *An algorithm for finding the zero crossing of time signals with Lipschitzeans derivatives*, Measurements, 16 (1995), pp. 37–49.

[7] ———, *Fast detection of the first zero-crossing in a measurement signal set*, Measurements, 19 (1996), pp. 29–39.

[8] R. Hammer, M. Hocks, U. Kulisch, and D. Ratz, *C++ Toolbox for Verified Computing*, Springer-Verlag, 1995.

[9] E. Hansen, *Global Optimization Using Interval Analysis*, vol. 165 of Pure and applied mathematics, Marcel Dekker, Inc, 1992.

[10] T. Ibaraki, *Theoretical comparisons of search strategies in branch and bound algorithms*, Int. J. Comput. Inform. Sci., 5 (1976), pp. 315–344.

[11] D. Kalra and A. H. Barr, *Guaranteed ray intersections with implicit surfaces*, Computer Graphics, 23 (1989), pp. 297–306.

[12] R. B. Kearfott, *Rigorous Global Search: Continuous Problems*, Kluwer Academic Publishers, 1996.

[13] ———, *Empirical evaluation of innovations in interval branch and bound algorithms for nonlinear systems*, SIAM Journal on Scientific Computing, 18 (1997), pp. 574–594.

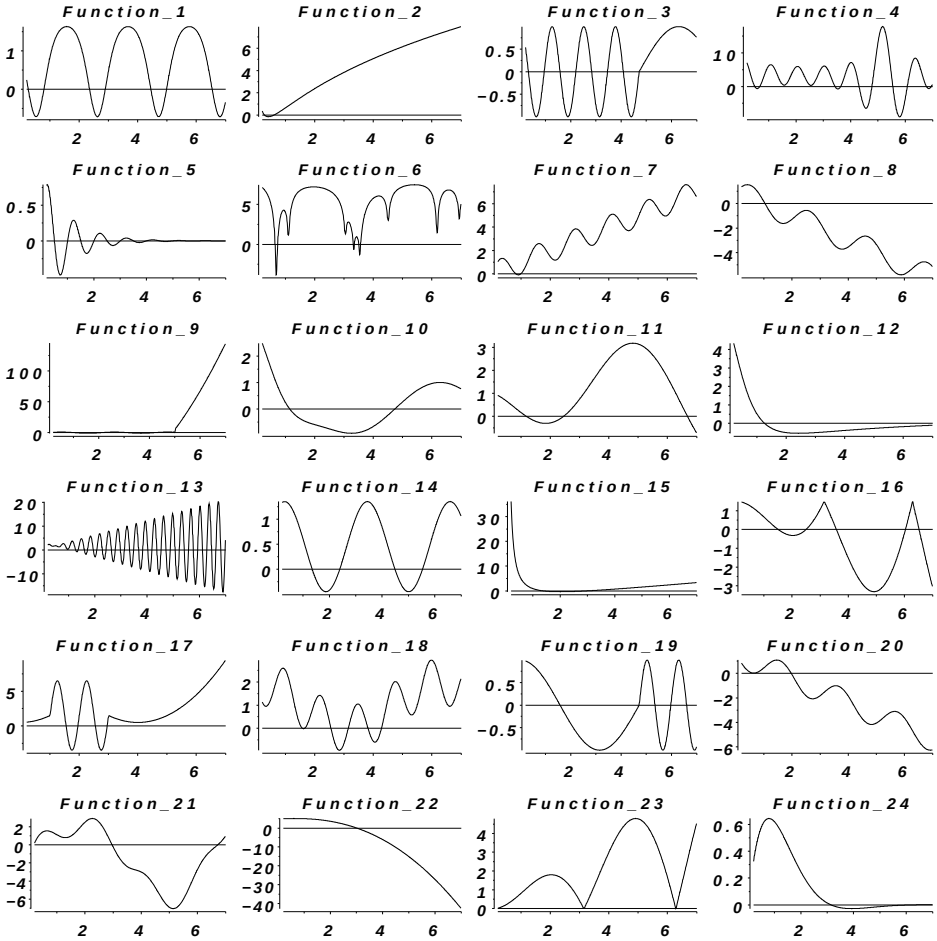
[14] O. Knüppel, *BIAS-Basic Interval Arithmetic Subroutines*, Tech. Report 93.3, University of Hamburg, 1993.

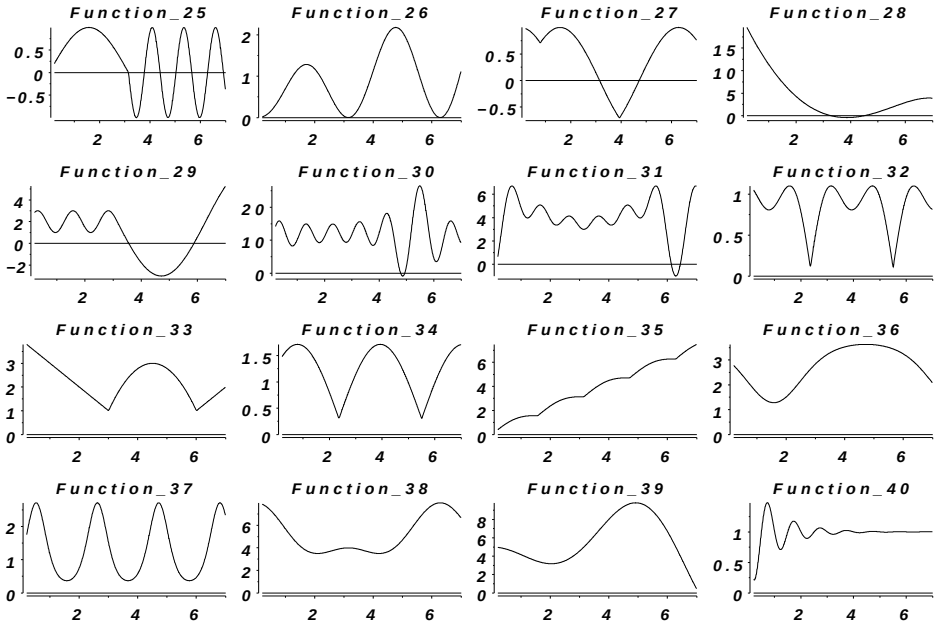
[15] K. Makino and M. Berz, *Higher Order Verified Inclusions of Multidimensional Systems by Taylor Models*, Nonlinear Analysis, in print, (2001). Available at http://bt.nscl.msu.edu/pub/.

[16] S. Mallat, *Zero crossing of a wavelet transform*, IEEE Transactions on Information Theory, 17 (1991), pp. 1019–1033.

[17] D. Mitchell, *Robust ray intersection with interval arithmetic*, in Graphics Interface'90, 1990, pp. 68–74.
[18] L. G. Mitten, *Branch and bound methods: general formulation and properties*, Operation Research, 18 (1970), pp. 24–34.
[19] R. Moore, *Interval analysis*, Prentice-Hall, 1966.
[20] A. Neumaier, *Interval Methods for Systems of Equations*, Cambridge University Press, Cambridge, 1990.
[21] W. Press, Y. Flanner, S. Teukolsky, and W. Vetterling, *Numerical Recipes: The art of Scientific Computing*, Cambridge University Press, 1986.
[22] H. Ratschek and J. Rokne, *New computer methods for global optimization*, Ellis Horwood Lmt.(John Willey & Sons), 1988.
[23] Y. D. Sergeyev, P. Daponte, D. Grimaldi, and A. Molinaro, *Two methods for solving optimization problems arising in electronic measurement and electrical engineering*, SIAM Journal on Optimization, 10 (1999), pp. 1–21.
[24] Q. Zhu, M. Kam, and R. Yeager, *Non-parametric identification of QAM constellations in noise*, in IEEE Conference on Acoustic, Speech and Signal Processing, vol. 4, 1993, pp. 184–187.

Appendix A. Graphs of test functions.




Function_25
Function_26
Function_27
Function_28
Function_29
Function_30
Function_31
Function_32
Function_33
Function_34
Function_35
Function_36
Function_37
Function_38
Function_39
Function_40