Bull. Austral. Math. Soc. **77** (2008), 353–363
doi:10.1017/S0004972708000117

AN ALGORITHM FOR FINDING ALL ZEROS OF VECTOR FUNCTIONS

IBRAHEEM ALOLYAN

(Received 10 May 2007)

Abstract

Computing a zero of a continuous function is an old and extensively researched problem in numerical computation. In this paper, we present an efficient subdivision algorithm for finding all real roots of a function in multiple variables. This algorithm is based on a simple computationally verifiable necessity test for the existence of a root in any compact set. Both theoretical analysis and numerical simulations demonstrate that the algorithm is very efficient and reliable. Convergence is shown and numerical examples are presented.

2000 *Mathematics subject classification*: 65H10, 65Y20.

Keywords and phrases: numerical solution of nonlinear system of equation, exclusion test, decomposable functions, computational complexity.

1. Introduction

One of the oldest and most basic problems in mathematics is that of solving an equation $F(x) = 0$. This problem has motivated many theoretical developments including the fact that solution formulas do not in general exist (as in the case where $F(x)$ is a fifth degree polynomial). Thus, the development of algorithms for finding solutions has historically been an important enterprise. Our goal here is to contribute to this enterprise by describing a numerical method for finding all solutions of an equation $F(x) = 0$ for a relatively broad class of nonlinear functions F. By all solutions, we mean all solutions lying in a given compact set $\mathscr{R}$. Essentially, the method starts with the compact set $\mathscr{R}$ and uses a recursive subdivision process to eliminate all of $\mathscr{R}$ except the zeros contained in $\mathscr{R}$.

Several different methods have been proposed for solving nonlinear systems. A class of methods that is suitable for finding curves of solutions, for example, in the case of bifurcation problems, is that of continuation methods [1, 13]. As the name implies, these methods piece together local problems, thus using local methods to solve a problem of a more global character. Of course, a known solution is needed to start

Supported by the research center project # (Math/2008/5).

the process. Homotopy algorithms [1, 12, 15, 18] are similar to continuation methods in that the idea is to start from a known solution and progress along a path to another solution. However, in the case of homotopy methods the path is the homotopy path that connects the given system to an artificial system that is readily solvable. Homotopy methods have proven to be effective in many cases, and in the case of polynomial systems have been incorporated into an algorithm for finding all the solutions of the system [12].

Numerical methods based on topological degree theory have also been proposed [16, 17]. In these methods it is the computation of the degree of a function on a given domain that is the major task. As with degree theory itself, a non-zero degree is only a sufficient condition for the existence of a zero in the given region. Thus, some solutions can remain undetected. For example, a pair of solutions in a given region can produce a degree of zero. Closely related to the topological degree methods are the search methods of Hsu [7, 8] which subdivide the domain into cells and then apply index theory to determine approximate locations of the zeros of a function. Another method of finding the solution of a nonlinear system of equations is the exclusion algorithms [2, 3, 5, 6, 19, 20].

Exclusion algorithms are a well-known tool in the area of interval analysis [4, 9–11] for finding all solutions of a system of nonlinear equations

$$F : \mathscr{R} \subset \mathbb{R}^n \to \mathbb{R}^n, \quad F(x) = 0, \quad F \in C \tag{1.1}$$

where $\mathscr{R}$ is a cell, that is a rectangular box, and $C = C(\mathscr{R})$ is the space of continuous functions.

These types of algorithms are different in principle from those of homotopy, interval and cell mapping dynamical analysis approaches [1]. They are based on cellular partitions of $\mathscr{R}$ and use of a computationally verifiable necessity test called the *root condition* for the solution of (1.1). The main advantages of the algorithms include their simplicity, reliability, and general applicability.

We now give some definitions that will be used in the subsequent sections. In $\mathbb{R}^n$, we use the component-wise $\leq$ as a partial ordering, that is we define $x \leq y$ if $x_i \leq y_i$ for all $i = 1, 2, \ldots, n$. A cell $X \subseteq \mathbb{R}^n$ is a rectangular box, that is there are two vectors $m_X, r_X \in \mathbb{R}^n$ with $r > 0$, such that

$$X = [m_X - r_X, m_X + r_X] = \{x \in \mathbb{R}^n : m_X - r_X \leq x \leq m_X + r_X\}.$$

We call $\underline{X} := m_X - r_X$ the lower corner, $\overline{X} := m_X + r_X$ the upper corner, m_X the midpoint, and r_X the radius of X. The mesh size of a cell X is defined to be $\|r_X\|$.

We briefly describe our view of an exclusion method. We begin with some region, e.g. a cell, in which we expect all zeros to be found. The algorithm is based on a given root condition which can be applied to each cell. If a cell fails the condition, we know it will not contain a zero and it can be discarded. If a cell satisfies the condition, it

https://doi.org/10.1017/S0004972708000117 Published online by Cambridge University Press

is subdivided and the condition is applied to the new cells. This leads to a recursive algorithm as we will show in the next section. The success and efficiency of these algorithms will strongly depend on the choice of the condition. This paper is organized as follows. In Section 2, a general framework of the algorithm is developed. In Section 3, a root condition is given with some examples. In Section 4, convergence and complexity analysis of the algorithms is presented. In Section 5, numerical simulation results are provided to demonstrate the effectiveness of the algorithms.

2. Basic version of the algorithm

In this section, we present some basic versions of an exclusion algorithm for finding all the zeros of a real-valued function $F(x)$ that lie in a compact set in $\mathbb{R}^n$. Since any compact set in $\mathbb{R}^n$ can be enclosed in a rectangle, we assume for simplicity that the given compact set is a closed rectangle $\mathscr{R} \in \mathbb{R}^n$, with sides parallel to coordinate planes. Thus, we present these algorithms in the context of finding the set of zeros in $\mathscr{R}$ of a system of equations $F(x) = 0$ (that is $Z := \{x \in \mathscr{R} : F(x) = 0\}$).

By an exclusion algorithm, we mean a process that generates a sequence of subsets $\{\mathscr{R}_i\}_{i=1}^{\infty}$ that approximate Z with successively increasing accuracy. By construction, this will be a nested sequence, $\mathscr{R}_{i+1} \subset \mathscr{R}_i$ for all i, that converges to Z in the sense that $Z := \bigcap_{i=1}^{\infty}\{\mathscr{R}_i\}$. In each case, the process consists of subdividing the current subset $\mathscr{R}_i$ into smaller subsets which are then either retained or discarded according to a selection criterion. Those that are retained determine the next subset $\mathscr{R}_{i+1}$. In this section, four different selection criteria are suggested. A sequence of partitions of $\mathscr{R}$ is needed for the subdivision process. A first partition of $\mathscr{R}$ into congruent subrectangles can be obtained by slicing $\mathscr{R}$ into two equal parts in one of the coordinate directions. For example, if $\mathscr{R}$ is a rectangle in the plane, this results in two congruent subrectangles if we consider the partition in the x_1-axis. For a rectangle in R^n there will also be two subrectangles.

If $\mathscr{S}$ is a finite set of cells, we say that $\mathscr{S}$ is a cellular partition of $\mathscr{R}$ if $\mathscr{R}$ is the union of the cells in $\mathscr{S}$, and the intersection of each two cells in $\mathscr{S}$ is either a common face or the empty set. If $\mathscr{S}_1$ and $\mathscr{S}_2$ are two cellular partitions of $\mathscr{R}$, we say that $\mathscr{S}_2$ is finer than $\mathscr{S}_1$ if each cell in $\mathscr{S}_2$ is contained in a cell in $\mathscr{S}_1$.

An exclusion algorithm systematically discards cells as it progresses. In order to do this, the algorithm makes use of some test which we will refer to as a root condition. A root condition is a necessary, but not sufficient, condition which must be satisfied if a zero point is present in a cell. Thus, if a cell fails the root condition, we know it does not contain a zero and may be discarded immediately. This is one of the desirable features of exclusion algorithms. Now we formally define a root condition.

DEFINITION 2.1 (Root condition). A root condition is a computationally verifiable necessity test for the presence of a zero in a cell.

https://doi.org/10.1017/S0004972708000117 Published online by Cambridge University Press

For example, when F is a Lipschitz function which satisfies the condition $\|F(x) - F(y)\| \leq L\|x - y\|$, where $L > 0$, then there is a solution x^* in $X \in \mathscr{R}$ only if $\|F(m_X)\| \leq L\|m_X - x^*\| \leq L\|r_X\|$. This implies that $\|F(m_X)\| \leq L\|r_X\|$ is a root condition for existence of solutions to $F(x) = 0$ in each cell of $\mathscr{R}$.

In order to locate the set of zeros Z, the algorithm starts with $\mathscr{R}$, then it is based on a given sequence of refining partitions $\mathscr{S}_i$ and uses at each stage a given root condition to exclude cells which cannot contain a root of F. The remaining cells are then stored in $\mathscr{R}_i$.

In summary, the exclusion algorithm consists of applying an exclusion test on a cell. If the test is negative, that is the necessary condition does not hold, then we are sure that the interval contains no zero, and we exclude it. Otherwise, we bisect the interval and apply the test again. This leads to the following algorithm.

ALGORITHM 2.1.

(1) Let $\mathscr{S}_i$ be a sequence of cellular partitions of $\mathscr{R}$ with $\mathscr{S}_0 = \{\mathscr{R}\}$, such that $\mathscr{S}_{i+1}$ is finer than $\mathscr{S}_i$.
(2) Let $\mathscr{R}_0 = \{\mathscr{R}\}$ (initialization)
(3) For $i = 0, 1, 2, \ldots$
    $\mathscr{R}_{i+1} := \phi$
    for $Y \in \mathscr{S}_{i+1}$ such that $Y \subset X \in \mathscr{R}_i$
        If Y satisfies the root condition,
            then $\mathscr{R}_{i+1} := \mathscr{R}_{i+1} \cup \{Y\}$.

Whenever one cycle of co-ordinate bisections is accomplished, we say that we have reached a new bisection level, and we think of an exclusion algorithm as performing a fixed number of bisection levels. The intervals which have not been discarded after i bisection levels will be considered as the intervals which the algorithm generates on the ith bisection level. This list of intervals is denoted by $\mathscr{R}_i$ in the algorithm. Obviously, if $\mathscr{R}_i = \phi$ for some level i, then the algorithm has shown that there are no zero points of F in the initial interval $\mathscr{R}$. It is clear that the efficiency of exclusion algorithms hinges mainly on the construction of a good exclusion test which is computationally inexpensive but relatively tight. Otherwise, too many intervals remain undiscarded on each bisection level, and this leads to significant numerical inefficiency. Note the preceding definition of the list $\mathscr{S}_i$ of intervals which were not discarded after the initial interval $\mathscr{R}$ has been bisected i times. Also note that the algorithm may stop with an empty list $\mathscr{S}_i$ indicating that F has no zero in $\mathscr{R}$. For clarity of exposition and notation, the list of intervals is processed breadth-first rather than depth-first.

3. The monotonicity root condition

In this section, we present a root condition that can be used to discard cells in Algorithm 2.1. We first define *isotone functions*.

https://doi.org/10.1017/S0004972708000117 Published online by Cambridge University Press

DEFINITION 3.1 (Isotone function). A function $F : \mathscr{R} \to \mathbb{R}^n$ is said to be isotone if $F(x) \leq F(y)$ whenever $x \leq y$.

DEFINITION 3.2 (Monotonically decomposable). The function F is said to be monotonically decomposable if there are two isotone functions G and H such that $F = G - H$.

The following examples illustrate monotonically decomposable functions.

EXAMPLE 3.1.

(1) Consider the function $F : [0, 10] \to \mathbb{R}$ which is defined by

$$F(x) = 3x^4 - 4x^3 - 5x + 9.$$

If we take $G(x) = 3x^4 + 9$ and $H(x) = 4x^3 + 5x$, then G and H are isotone and $F(x) = G(x) - H(x)$.

(2) Consider the function $F : [-2, 4] \times [-1, 5] \to \mathbb{R}^2$ which is defined by

$$F(x_1, x_2) = \begin{bmatrix} x_1^2 - x_2^2 \\ 2x_1x_2 + x_1 \end{bmatrix}.$$

We can expand this function about the point $(-2, -2)$, and we get

$$G(x_1, x_2) = \begin{bmatrix} x_1^2 + 4x_1 + 2x_2 + 3 \\ 2x_1x_2 + 3x_1 + 4x_2 + 4 \end{bmatrix},$$
$$H(x_1, x_2) = \begin{bmatrix} x_2^2 + 4x_1 + 2x_2 + 3 \\ x_1 + 4x_2 + 4 \end{bmatrix}.$$

One can easily check that H and G are isotone on the cell $[-2, 4] \times [-1, 5]$ and $F(x) = G(x) - H(x)$.

THEOREM 3.1. *Let F be a monotonically decomposable function (that is $F = G - H$, where G and H are isotone functions), and let X be a cell that contains a solution to the system (1.1). Then we have the monotonicity root condition*

$$G(\underline{X}) \leq H(\overline{X}) \quad \textit{and} \quad H(\underline{X}) \leq G(\overline{X}). \tag{3.1}$$

PROOF. If F contains a solution, say x^*, to the system $F(x) = 0$, then we have

$$G(x^*) - H(x^*) = F(x^*) = 0.$$

By the isotone property of G and H, we have

$$G(\underline{X}) - H(\overline{X}) \leq 0 \quad \text{and} \quad H(\underline{X}) - G(\overline{X}) \leq 0.$$

From these equations, we obtain the root condition (3.1). □

https://doi.org/10.1017/S0004972708000117 Published online by Cambridge University Press

4. Convergence and complexity

In this section, we prove the main result underlying our algorithm. We prove both the global convergence of Algorithm 2.1 and estimate the complexity of the algorithm. The sequence we obtain using Algorithm 2.1 does indeed converge to the roots of F (namely, Z).

THEOREM 4.1. *Let $\mathscr{R}_i$ be the set of cells generated by Algorithm 2.1 using the monotonicity root condition (3.1), and let Z be the set of solutions to the system (1.1). If $F = G - H$ is monotonically decomposable with G and H continuous, then $Z \neq \phi$ if and only if the algorithm does not terminate in a finite number of steps, and we have*

$$\lim_{i\to\infty} \mathscr{R}_i = Z.$$

PROOF. Let $\widetilde{\mathscr{R}} := \lim_{i\to\infty} \mathscr{R}_i$, so $\widetilde{\mathscr{R}} \neq \phi$ because the algorithm does not terminate in a finite number of steps. Since each $\mathscr{R}_i$ contains all roots of F, we have $Z \subseteq \widetilde{\mathscr{R}}$. Now, for any $\widetilde{x} \in \widetilde{\mathscr{R}}$, there is a sequence of cells $\{X_i\}$, where $X_i \in \mathscr{R}_i$, such that $\widetilde{x} \in X_i$ for all $i \geq 1$.

Now for each cell X_i, it must satisfy the monotonicity root condition (3.1), and we have

$$\|(G+H)(\overline{X_i}) - (G+H)(\underline{X_i})\| = \|G(\overline{X_i}) - H(\underline{X_i})\| + \|H(\overline{X_i}) - G(\underline{X_i})\|.$$

Since $G + H$ is uniformly continuous on X_i, we have

$$\|(G+H)(\overline{X_i}) - (G+H)(\underline{X_i})\| \to 0 \quad \text{as } \|r_{X_i}\| \to 0.$$

Therefore, we have

$$\|G(\overline{X_i}) - H(\underline{X_i})\| \to 0 \quad \text{and} \quad \|H(\overline{X_i}) - G(\underline{X_i})\| \to 0 \quad \text{as } i \to \infty.$$

Now we show that $\widetilde{x} \in Z$. If $F(\widetilde{x}) = G(\widetilde{x}) - H(\widetilde{x}) \geq 0$, then

$$0 \leq \|G(\widetilde{x}) - H(\widetilde{x})\| \leq \|G(\overline{X_i}) - H(\underline{X_i})\| \to 0 \quad \text{as } i \to \infty.$$

If $F(\widetilde{x}) = G(\widetilde{x}) - H(\widetilde{x}) \leq 0$, then

$$0 \leq \|G(\widetilde{x}) - H(\widetilde{x})\| \leq \|H(\overline{X_i}) - G(\underline{X_i})\| \to 0 \quad \text{as } i \to \infty.$$

Thus, $F(\widetilde{x}) = 0$ and $\widetilde{x} \in Z$, so $\widetilde{\mathscr{R}} \subseteq Z$; therefore, $\widetilde{\mathscr{R}} = Z$. □

In the following theorem, we show that the number of cells that cannot be excluded by the root condition (3.1) does not go to infinity. This means that in Algorithm 2.1 there should be a positive constant K such that the number of cells in $\mathscr{R}_i$ ($|\mathscr{R}_i|$) for $i = 1, 2, \ldots$ does not exceed K. This is imperative for exclusion algorithms to be successful; otherwise, the cells that need to be checked by the algorithm increase greatly so that the algorithm is hardly ever able to reach a pre-assigned accuracy.

https://doi.org/10.1017/S0004972708000117 Published online by Cambridge University Press

THEOREM 4.2. *Let $\mathscr{R}_i$ be the sets of cells generated by Algorithm 2.1 using the monotonicity root condition (1.1), and let Z be the set of solutions to the system. If G and H are Lipschitz functions and Z consists of a finite number of regular solutions to the system (namely, $\|F'(\widetilde{x})\|$ is invertible for any $\widetilde{x} \in Z$), then there is a constant $K > 0$ such that $|\mathscr{R}_i| \le K$ for all $i \ge 1$.*

PROOF. Since G and H are Lipschitz functions, $G + H$ is also a Lipschitz function; therefore, there is a constant L such that

$$\|(G + H)(x) - (G + H)(y)\| \le L\|x - y\| \quad \text{for all } x, y \in \mathscr{R}.$$

Now consider any cell $X \in \mathscr{R}$, then either $F(m_X) = G(m_X) - H(m_X) \ge 0$ or $F(m_X) = G(m_X) - H(m_X) \le 0$. In the first case, we have $0 \le G(m_X) - H(m_X) \le G(\overline{X}) - H(\underline{X})$. We combine this condition with the monotonicity condition $H(\overline{X}) - G(\underline{X}) \ge 0$, and we get

$$0 \le \|F(m_X)\| \le \|(G + H)(\overline{X}) - (G + H)(\underline{X})\|.$$

In the second case, we can show that we will get the same result. Therefore, we have

$$\|F(m_X)\| \le \|(G + H)(\overline{X}) - (G + H)(\underline{X})\| \le L\|\overline{X} - \underline{X}\| \le 2L\|r_X\|. \quad (4.1)$$

It is clear that if $\widetilde{x}$ lies on the cell X then $\|m_X - \widetilde{x}\| \le \|r_X\|$. Now consider the set

$$\overline{\mathscr{R}} := \{X \in \mathscr{R} : \|m_X - \widetilde{x}\| \ge \|r_X\| \text{ for all } \widetilde{x} \in Z\}.$$

Since F is continuous on the compact set $\overline{\mathscr{R}}$, $\|F(x)\|$ attains a minimum $m > 0$ on $\overline{\mathscr{R}}$ (that is $\|F(x)\| \ge m$ for all $x \in \overline{\mathscr{R}}$). Now consider $m_X \in \overline{\mathscr{R}}$, and take $\|r_X\|$ sufficiently small such that $2L\|r_X\| < m < \|F(m_X)\|$. Therefore, the cell X does not satisfy (4.1) and thus it will be discarded.

Therefore, for sufficiently large i, the volume of a ball around $\widetilde{x}$ which may contain a cell that satisfies the root condition in $\mathbb{R}^n$ is at most $((2L + 1)2\|r_X\|)^n$. Thus the number of cells that could fit inside the ball is approximately $(2L + 1)^n$. Since we are dealing with a finite number of roots M, the total number of cells will be bounded by $(2L + 1)^n M = K$. □

The functions G and H in condition (3.1) are computed globally (that is they are isotone on the interval $\mathscr{R}$). If we consider the functions G_k and H_k to be isotone on the subcell $X_k \subset \mathscr{R}$, then we will get tighter conditions and more cells will be discarded which improves the efficiency of the algorithm. One can verify the following theorem easily.

THEOREM 4.3. *If the function F is a polynomial then F is monotonically decomposable on any interval $\mathscr{R}$. Let $\mathscr{S} \subset \mathscr{R}$ be a subinterval. If we expand the function F about the point $\underline{\mathscr{R}}$ and take the positive and negative coefficients to be $G_{\mathscr{R}}$ and $H_{\mathscr{R}}$, respectively, and we expand F about the point $\underline{\mathscr{S}}$ and take the positive and negative coefficients to be $G_{\mathscr{S}}$ and $H_{\mathscr{S}}$, respectively, then we have*

$$G_{\mathscr{R}}(x) \le G_{\mathscr{S}}(x) \quad \text{and} \quad H_{\mathscr{R}}(x) \le H_{\mathscr{S}}(x) \quad \text{for all } x \in \mathscr{S}.$$

https://doi.org/10.1017/S0004972708000117 Published online by Cambridge University Press

Here is an example to illustrate the previous theorem.

EXAMPLE 4.1. Consider the function $F : \mathscr{R} := [-2, 2] \times [-1, 1] \to \mathbb{R}^2$ which is defined by

$$F(x_1, x_2) = \begin{bmatrix} x_1^2 - x_2^2 \\ 2x_1x_2 + x_1 \end{bmatrix}.$$

We may take the global monotonicity functions $G_{\mathscr{R}}$ and $H_{\mathscr{R}}$ to be

$$G(x_1, x_2) = \begin{bmatrix} (x_1 + 2)^2 + 2(x_2 + 1) + 3 \\ 2(x_1 + 2)\ (x_2 + 1) + (x_1 + 2) \end{bmatrix},$$
$$H(x_1, x_2) = \begin{bmatrix} 4(x_1 + 2) + (x_2 + 1)^2 \\ 2(x_1 + 2) + 4(x_2 + 1) + 2 \end{bmatrix}.$$

One can easily check that H and G are isotone on the interval $\mathscr{R}$ and $F(x) = G(x) - H(x)$. If we consider the partition of $\mathscr{R}$ to be

$$\mathscr{R} = \bigcup_{i=1:4} X_i,$$

where

$$X_1 = [-2, 0] \times [-1, 0], \quad X_2 = [0, 2] \times [-1, 0],$$
$$X_3 = [-2, 0] \times [0, 1], \quad X_4 = [0, 2] \times [0, 1],$$

then we can consider the local monotonicity functions G_i and H_i for $i = 1, \ldots, 4$ to be

$$G(x_1, x_2) = \begin{bmatrix} (x_1 + 2)^2 + 2(x_2 + 1) + 3 \\ 2(x_1 + 2)\ (x_2 + 1) + (x_1 + 2) \end{bmatrix},$$
$$H(x_1, x_2) = \begin{bmatrix} 4(x_1 + 2) + (x_2 + 1)^2 \\ 2(x_1 + 2) + 4(x_2 + 1) + 2 \end{bmatrix}.$$

on the interval X_1,

$$G_2(x_1, x_2) = \begin{bmatrix} x_1^2 + 2(x_2 + 1) \\ 2x_1(x_2 + 1) \end{bmatrix}, \quad H_2(x_1, x_2) = \begin{bmatrix} (x_2 + 1)^2 + 1 \\ x_1 \end{bmatrix},$$

on the interval X_2,

$$G_3(x_1, x_2) = \begin{bmatrix} (x_1 + 2)^2 + 4 \\ 2(x_1 + 2)x_2 + (x_1 + 2) \end{bmatrix}, \quad H_3(x_1, x_2) = \begin{bmatrix} 4(x_1 + 2) + x_2^2 \\ 4x_2 + 2 \end{bmatrix},$$

on the interval X_3,

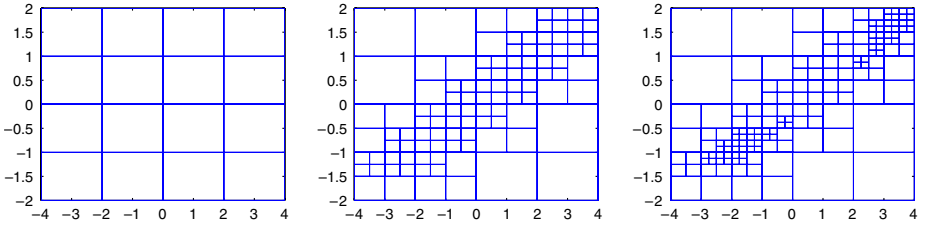
$$G_4(x_1, x_2) = \begin{bmatrix} x_1^2 \\ 2x_1x_2 + x_1 \end{bmatrix}, \quad H_4(x_1, x_2) = \begin{bmatrix} x_2^2 \\ 0 \end{bmatrix},$$

on the interval X_4. One can easily check that $F(x) = G_i(x) - H_i(x)$ on the interval X_i, and G_i and H_i are isotone on the interval X_i.

We can note that

$$G_i(x) \le G(x) \quad \text{and} \quad H_i(x) \le H(x) \quad \text{for all } x \in X_i.$$

https://doi.org/10.1017/S0004972708000117 Published online by Cambridge University Press



(a) level 2 (b) level 4 (c) level 5

FIGURE 1. Partitions of exclusion algorithm at level i.

5. Numerical experiments

The application of an exclusion algorithm to an example problem is discussed in this section. We present a numerical example to find all solutions of $F(x) = 0$ in a given interval $\mathscr{R} \in \mathbb{R}$. This example investigates the performance of the algorithm in solving the following problem that was proposed by the author. We will graphical present the set $\mathscr{R}_i$ for some values of i to show the partitions of the domain $\mathscr{R}$ in the level i. We will then present a table where we list the number of non-discarded intervals on the bisection level i. In order to find the roots of the following function, we use Algorithm 2.1 which has been implemented in MATLAB [14] and the experiments executed on a PC with a Pentium Centrino 2000 MHz processor.

EXAMPLE 5.1. To convey a sense of how the algorithm works, we consider the function $F : \mathscr{R} \to \mathbb{R}^2$, where $\mathscr{R} := [-4, 4] \times [-2, 2]$, defined by

$$F(x_1, x_2) = \begin{bmatrix} x_1 - 2x_2 \\ x_1x_2 + x_1 - 4x_2 - 4 \end{bmatrix}.$$

This is a simple polynomial vector field having two regular zeros $(-2, -1)$ and $(4, 2)$.

We can define the global monotonicity functions G and H to be

$$G(x_1, x_2) = \begin{bmatrix} x_1 \\ x_1x_2 + 3x_1 + 4x_2 + 8 \end{bmatrix}, \quad H(x_1, x_2) = \begin{bmatrix} 2x_2 \\ 2x_1 + 8x_2 + 12 \end{bmatrix}.$$

We now apply Algorithm 2.1 to the function F using both the global and local monotonicity functions with Tolerance, $\mathrm{Tol} = 0.1,\ 0.01,\ 0.001$, and 0.0001. In Figure 1, we show the partitions of the domain $\mathscr{R}$ in the level i and we note that more partitions are performed near the roots of the function. The number of cells in $\mathscr{R}$, and the number of *root condition checks* (RC) is shown in Table 1.

In Table 1, we note that the number of cells when we use the global monotonicity functions (MF) is more than the number of cells in the local case. Finally, we plot the final refinement of Algorithm 2.1 in Figure 2.

https://doi.org/10.1017/S0004972708000117 Published online by Cambridge University Press

TABLE 1. Approximation of the roots of F.

Tol	Level	x^*	Global MF		Local MF	
			$\lvert\mathscr{R}\rvert$	RC	$\lvert\mathscr{R}\rvert$	RC
0.1	7	(3.9688, 1.9844), (−2.0313, −0.9531)	31	235	23	147
0.01	10	(3.9961, 1.9980), (−2.0039, −0.9941)	43	557	27	346
0.001	13	(3.9995, 1.9998), (−2.0005, −0.9993)	47	784	32	478
0.0001	17	(4.0000, 2.0000), (−2.0000, −1.0000)	45	1033	29	703

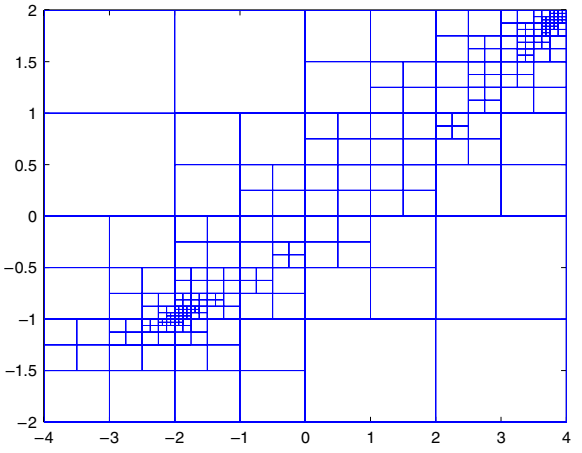


FIGURE 2. Applying the exclusion algorithm to Example 5.1.

6. Conclusion

In this paper we have introduced a technique to solve a system of nonlinear equations. We have formulated an exclusion test for Algorithm 2.1 for a monotonicity decomposable function. It has been shown that the algorithm converges to the roots of F and the number of cells, as Algorithm 2.1 proceeds, is bounded. Finally, the numerical implementation of the obtained methods has been considered. The family of iterative methods that are introduced in this paper uses a multi-precision and adaptive floating point arithmetic with low computing times. Our numerical results indicate that Algorithm 2.1 with global monotonicity functions converges to the roots much faster when the monotonicity functions are chosen locally.

https://doi.org/10.1017/S0004972708000117 Published online by Cambridge University Press

References

[1] E. Allgower and K. Georg, *Introduction to Numerical Continuation Methods* (SIAM, Philadelphia, PA, 2003).

[2] E. Allgower, M. Erdmann and K. Georg, 'On the complexity of exclusion algorithms for optimization', *J. Complexity* **18** (2002), 573–588.

[3] I. Alolyan, 'A new exclusion test for finding the global minimum', *J. Comput. Appl. Math.* **200**(2) (2007), 491–502.

[4] T. Csendes and D. Ratz, 'Subdivision direction selection in interval methods for global optimization', *SIAM J. Numer. Anal.* **34**(3) (1997), 922–938.

[5] K. Georg, 'A new exclusion test', *Proc. Int. Conf. on Recent Advances in Computational Mathematics*, *J. Comput. Appl. Math.* **152**(1–2) (2003), 147–160.

[6] K. Georg, 'Improving the efficiency of exclusion algorithms', *Adv. Geom.* **1** (2001), 193–210.

[7] C. S. Hsu and R. S. Guttalu, 'Index evaluation for dynamical systems and its application to locating all of the zeros of a vector function', *J. Appl. Mech.* **50** (1983), 858–862.

[8] C. S. Hsu and W. H. Zhu, 'A simplical mapping method for locating the zeros of a function', *Quart. Appl. Math.* **42** (1984), 41–59.

[9] R. Kearfott, 'Rigorous global search: continuous problems', in: *Nonconvex Optimization and its Applications*, Vol. 13 (Kluwer Academic, Dordrecht, 1996).

[10] R. B. Kearfott, 'Empirical evaluation of innovations in interval branch and bound algorithms for nonlinear algebraic systems', *SIAM J. Sci. Comput.* **18**(2) (1997), 574–594.

[11] R. Moore, *Methods and Applications of Interval Analysis*, SIAM Studies in Applied Mathematics, 2 (SIAM, Philadelphia, PA, 1979).

[12] A. P. Morgan, A. J. Sommese and L. T. Watson, 'Finding all isolated solutions to polynomial systems using HOMPACK', *ACM Trans. Math. Software* **15** (1989), 93–122.

[13] W. C. Rheinboldt, *Numerical Analysis of Parametrized Nonlinear Equations* (Wiley, New York, 1986).

[14] *Using MATLAB* (The MathWorks Inc. Natick, MA, 1996).

[15] M. Sosonkina, L. T. Watson and D. E. Stewart, 'A note on the end game in homotopy zero curve tracking', *ACM Trans. Math. Software* **22** (1996), 281–287.

[16] M. N. Vrahatis and K. L. Iordanidis, 'A rapid generalized method of bisection for solving systems of non-linear equations', *Numer. Math.* **49** (1986), 123–138.

[17] M. N. Vrahatis, 'Solving systems of nonlinear equations using the nonzero value of the topological degree', *ACM Trans. Math. Software* **14** (1988), 312–328.

[18] L. T. Watson, 'Globally convergent homotopy methods: A tutorial', *Appl. Math. Comput.* **31** (1989), 369–396.

[19] Z.-B. Xu, J.-S. Zhang and Y.-W. Leung, 'A general CDC formulation for specializing the cell exclusion algorithms of finding all zeros of vector functions', *Appl. Math. Comput.* **86** (1997), 235–259.

[20] J. C. Yakoubsohn, 'Numerical analysis of a bisection-exclusion method to find the zeros of univariant analytic functions', *J. Complexity* **21** (2005), 651–772.

IBRAHEEM ALOLYAN, Mathematics Department, College of Science,
King Saud University, PO Box 2455, Riyadh 11451, Saudi Arabia
e-mail: ialolyan05@yahoo.com

https://doi.org/10.1017/S0004972708000117 Published online by Cambridge University Press