
ELSEVIER

Journal of Computational and Applied Mathematics 137 (2001) 293–315

JOURNAL OF COMPUTATIONAL AND APPLIED MATHEMATICS

www.elsevier.com/locate/cam

An algorithm for finding all solutions of a nonlinear system

Michael W. Smiley[a,*], Changbum Chun[b]

[a] *Department of Mathematics, Iowa State University, 400 Carver Hall, Ames, IA 50011-2064, USA*

[b] *Department of Mathematics, Seoul National University, Seoul, South Korea*

Received 29 November 1999; received in revised form 12 November 2000

Abstract

Let $f:X \to \mathbb{R}^k$ be a Lipschitz continuous function on a compact subset $X \subset \mathbb{R}^d$. Subdivision algorithms are described that can be used to find all solutions of the equation $f(x)=0$ that lie in X. Convergence is shown and numerical examples are presented. Modifications of the basic algorithm which speed convergence are given for the case of nondegenerate zeros of a vector field.

MSC: 65H10; 68Q25

Keywords: Subdivision algorithm; Newton's method

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 X. Essentially, the method starts with the compact set X and uses a recursive subdivision process to eliminate all of X but the zeros contained in X. In this sense, it is a variant of Weyl's algorithm [20] for finding the zeros of a polynomial (see [11] for a description of Weyl's algorithm).

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

* Corresponding author.

E-mail address: mwsmiley@iastate.edu (M.W. Smiley).

0377-0427/01/$ - see front matter
PII: S0377-0427(00)00711-1

that of continuation methods (cf. [1,12]). 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 the process. Homotopy algorithms (cf. [1,10,14,19]) 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 (cf. [10]).

Numerical methods based on topological degree theory have also been proposed (cf. [6,8,15,17,18]). 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 nonzero 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 [6,7] which subdivide the domain into cells and then apply simplicial index theory to determine approximate locations of the zeros of a function. Once approximate locations are determined, Newton-like methods can be used to refine the approximation.

Weyl's algorithm for solving a single polynomial equation (cf. [11,20]) seems to stand alone in its ability to reliably locate all solutions of the given equation. The algorithm uses a search and deletion algorithm starting with an initial square containing all the zeros of the polynomial. The square is partitioned into four congruent subsquares and a "*proximity test*", gauging the closeness of a midpoint to the set of zeros, is performed on each square. If the test guarantees that a subsquare cannot contain any of the zeros it is discarded. This process is then applied recursively to the subsquares not discarded. After the search region has been sufficiently refined, rapidly converging iterative techniques can be used to accelerate convergence.

Weyl's algorithm is an example of a subdivision algorithm. Subdivision algorithms have appeared in other contexts. For example, Gregory and his co-workers [3,4] have developed subdivision algorithms for the generation of curves and surfaces in geometric design problems, and Dellnitz and Hohmann [2] have used a subdivision algorithm to determine the attractor of a dynamical system. Some of the strengths of subdivision algorithms are their simplicity, reliability and robustness. For example, in the algorithms presented here for finding zeros, little is required of the function f. In principle, the zero set can be arbitrary, although there could be complexity issues in practice. A weakness of subdivision algorithms, in addition to complexity issues, is that it behaves like the bisection method. However, it is possible to accelerate the convergence. For example, to locate non-degenerate zeros of a C^1 vector field, Newton's method can be used as part of the process.

Although our primary interest is the problem of finding the zeros of a vector field, the algorithm is initially described for a real-valued function defined on a rectangle in $\mathbb{R}^d$. This is done in Section 2. When the problem is to find the zeros of a vector field, there are techniques that can be used to accelerate the convergence of the algorithm. These are described in Section 4 after a discussion in Section 3 on strategies for handling rectangles containing known zeros. Section 5 presents several example problems and results obtained using the composite subdivision algorithm described in Section 4. Section 6 gives proofs of convergence for the subdivision algorithms that appear in the paper.

2. Basic versions of the algorithm

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

Throughout it is assumed that F is Lipschitz continuous on $\mathscr{R}$. As such we may further assume $F: \mathscr{R} \to [0, \infty)$, by replacing F by $|F|$ if necessary. An important example is $F(x) = \|f(x)\|$, where $f(x)$ is a C^1 vector field on $\mathbb{R}^d$ and $\|\cdot\|$ is any norm on $\mathbb{R}^d$. Two natural choices in our work are the l_∞ norm, $\|x\|_\infty = \max\{|x_i|: 1 \leqslant i \leqslant d\}$, and the l_2 (or Euclidean) norm $\|x\|_2 = (\sum |x_i|^2)^{1/2}$. To measure distances within $\mathscr{R}$ we will always use the l_∞ norm.

By a subdivision algorithm for determining Z we mean a process that generates a sequence of subsets $\{X_i\}_{i=0}^{\infty}$ of $\mathscr{R}$ that approximate Z with successively increasing accuracy. By construction this will be a nested sequence, $X_{i+1} \subset X_i$ for all i, that converges to Z in the sense that $Z = \bigcap\{X_i: i \geqslant 0\}$. In each case, the process consists of subdividing the current subset X_i into smaller subsets which are then either retained or discarded according to a selection criterion. The ones that are retained determined the next subset X_{i+1}. In this section, four different selection criteria are suggested. Four closely related but different algorithms result.

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 each coordinate direction. For example, if $\mathscr{R}$ is a rectangle in the plane, this results in four congruent subrectangles. For a rectangle in $\mathbb{R}^d$ there will be 2^d subrectangles. In this context, it is natural to call $\mathscr{R}$ the parent rectangle and the congruent subrectangles the children of $\mathscr{R}$.

A sequence of partitions of $\mathscr{R}$ is defined inductively as follows. Let $\mathscr{P}_0 = \{\mathscr{R}\}$ be the trivial partition. Assuming the partition $\mathscr{P}_{i-1} = \{R_{i-1,j'}: 1 \leqslant j' \leqslant n_{i-1}\}$ has been defined, the partition $\mathscr{P}_i = \{R_{ij}: 1 \leqslant j \leqslant n_i\}$ is defined as the collection of all subrectangles R_{ij} obtained by applying the process described above to each parent rectangle $R_{i-1,j'} \in \mathscr{P}_{i-1}$. Since this is obviously a tree-like structure, the original rectangle will be called the root rectangle. A simple induction argument shows that the number of rectangles in $\mathscr{P}_i$ is $n_i = 2^{di}$. In fact, an indexing scheme that can be used to conveniently keep track of all the subrectangles generated in this way can be derived and is given in the appendix.

The first subdivision algorithm we consider is based on a selection criterion that uses minimum values. For each integer $i \geqslant 1$ define the subset of indices

$$J_i = \left\{ j \in \{1, 2, \ldots, n_i\}: \min_{x \in R_{ij}} F(x) \leqslant 2^{-i} \right\}. \tag{2.1}$$

If $j \in J_i$ then R_{ij} will be retained; otherwise it will be discarded. Thus, we define $X_i = \bigcup\{R_{ij}: j \in J_i\} \subset \mathscr{R}$. The basic idea is to discard any subrectangle R_{ij} which cannot contain a zero of F. This is contained in the definition of J_i: if $j \notin J_i$ then $F(x) \geqslant 2^{-i} > 0$ on R_{ij}. Obviously, $Z \subset X_i$ since no rectangle containing a zero can be discarded.

The sequence of subsets obtained by using this process is a nested sequence that converges to Z. The nested property is easy to verify. Suppose $j \in J_i$. By the geometric subdivision process there is

another index j' such that $R_{ij} \subset R_{i-1,j'}$ and

$$\min_{x \in R_{i-1,j'}} F(x) \leqslant \min_{x \in R_{ij}} F(x) \leqslant 2^{-i} < 2^{-(i-1)}.$$

Thus, $j' \in J_{i-1}$ and $R_{ij} \subset R_{i-1,j'} \subset X_{i-1}$. Hence, the definition above is equivalent to the inductive definition

$$J_i = \left\{ j \in \{1,2,\ldots,n_i\} \colon R_{ij} \subset X_{i-1} \text{ and } \min_{x \in R_{ij}} F(x) \leqslant 2^{-i} \right\},$$

where $X_0 = \mathscr{R}$ by convention. The convergence, $Z = \bigcap \{X_i \colon i \geqslant 0\}$, is shown in Section 6.

The naturally recursive nature of the geometric subdivision process is inherited by this zero finding algorithm. Suppose that $\mathscr{R}$ is a rectangle in the plane. Then to determine X_1, we first subdivide $\mathscr{R}$ into four subrectangles $R_{11}, R_{12}, R_{13}, R_{14}$. The selection criterion must be applied to each of these subrectangles. If the conclusion after testing R_{11} is that it is to be retained then it becomes in effect a new root rectangle. None of the remaining three rectangles have any effect on the retention or rejection of any of the children of R_{11}. Therefore, applying the subdivision algorithm to $\mathscr{R}$ is the sum of the results of applying the subdivision algorithm to each of the children of $\mathscr{R}$.

The subdivision algorithm based on the selection criterion (2.1) is impractical since the minimum of F over each relevant subset R_{ij} must be found. However, the Lipschitz property of F can be used to modify the selection criterion into a more realistic form. Let L denote the Lipschitz constant of F, x_{ij} denote the midpoint of the subrectangle R_{ij}, and ρ_i denote the radius of R_{ij} which is independent of j by congruency. (Since the l_∞ norm is being used ρ_i is half the maximum side length.) Set $X_0^{\text{lip}} = \mathscr{R}$, and inductively define for each integer $i \geqslant 1$ the set of indices

$$J_i^{\text{lip}} = \{ j \in \{1,2,\ldots,n_i\} \colon R_{ij} \subset X_{i-1}^{\text{lip}} \text{ and } F(x_{ij}) \leqslant 2^{-i} + \rho_i L \} \tag{2.2}$$

and the subset $X_i^{\text{lip}} = \bigcup \{ R_{ij} \colon j \in J_i^{\text{lip}} \}$. As in the first case, it can again be shown that the sequence $\{X_i^{\text{lip}}\}_{i=0}^{\infty}$ is nested with $X_\infty^{\text{lip}} = \{x \in \mathscr{R} \colon F(x) = 0\}$, where X_∞^{lip} is the intersection over all the X_i^{lip}'s.

Example 2.1. To convey a sense of how the algorithm works, we consider the (Euclidean) distance function $F(x) = \text{dist}(x, \mathscr{S})$, where $x \in \mathbb{R}^2$ and $\mathscr{S} = \mathscr{C} \cup \mathscr{D}$ is the union of a curve and a disk in the plane, with $\mathscr{C}$ and $\mathscr{D}$ chosen to be

$$\mathscr{C} = \{(\sin t, \sin 2t) \colon 0 \leqslant t \leqslant 2\pi\}, \quad \mathscr{D} = \{(x_1, x_2) \colon |x_1 - 0.5|^2 + x_2^2 \leqslant 0.1\}.$$

As with any distance function $F(x)$ is Lipschitz continuous with $L=1$. Clearly $\{x \in \mathbb{R}^2 \colon F(x)=0\} = \mathscr{S}$. Choosing $\mathscr{R} = [-2,2] \times [-2,2]$ and applying the algorithm determined by (2.2) produces a sequence of subsets that converges to $\mathscr{S}$. Fig. 1 contains four plots showing the subsets X_i^{lip} obtained when $i = 5, 6, 7, 8$. These sets were computed by using a recursive procedure that accepted a rectangle and a recursion depth as parameters. The procedure applied the selection criterion (2.2) to each of the four subrectangles obtained through subdivision. The retained subrectangles were passed recursively to the procedure with the recursion depth parameter increased by one, until the maximum recursion level i was reached. On the maximum level, the retained subrectangles were saved for printing and the procedure returned to the previous call terminating the recursion.

Although the subdivision algorithm based on the selection rule (2.2) is more practical than the one based on (2.1), it is slower to converge in the sense that $X_i \subset X_i^{\text{lip}}$ for each i. There are other

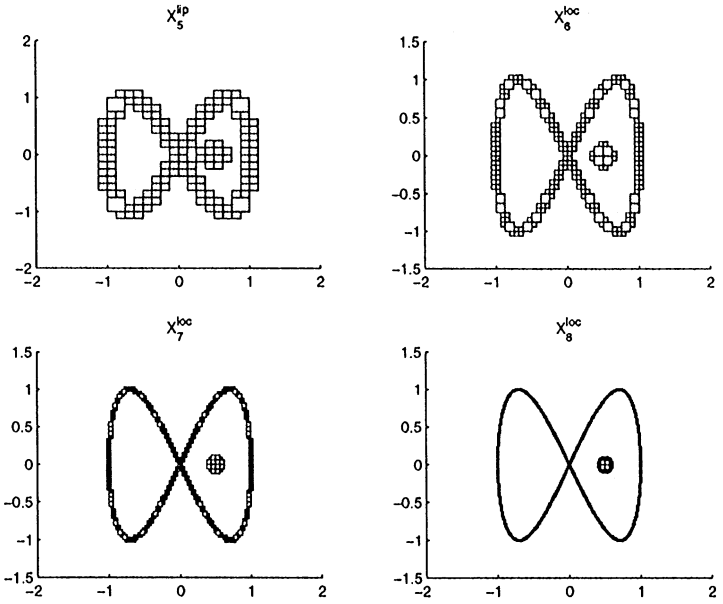

Fig. 1. The sets X_5^{lip}, X_6^{lip}, X_7^{lip}, X_8^{lip} are the unions of the shaded rectangles in the respective plots.

variants of the selection rule that regain some of the speed of convergence. The simplest one uses the local Lipschitz constants

$$L_{ij} = \sup_{x,y\in R_{ij},\ x\neq y} \frac{|F(x)-F(y)|}{\|x-y\|_\infty}. \tag{2.3}$$

With x_{ij} and ρ_i defined as before, set $X_0^{\text{loclip}} = \mathscr{R}$, and inductively define for each integer $i \geqslant 1$ the set of indices

$$J_i^{\text{loclip}} = \{j \in \{1,2,\ldots,n_i\}\colon R_{ij} \subset X_{i-1}^{\text{loclip}} \text{ and } F(x_{ij}) \leqslant 2^{-i} + \rho_i L_{ij}\} \tag{2.4}$$

and the subset $X_i^{\text{loclip}} = \bigcup\{R_{ij}\colon j \in J_i^{\text{loclip}}\}$. An easy induction argument shows that $J_i \subset J_i^{\text{loclip}} \subset J_i^{\text{lip}}$ and $X_i \subset X_i^{\text{loclip}} \subset X_i^{\text{lip}}$, so that the convergence of this variant of the algorithm is intermediate to the first two versions. In fact, the difference in convergence between the algorithm based on (2.2) and the algorithm based on (2.4) can be rather surprising. This is illustrated in Example 2.2.

However, to put this in a proper perspective the amount of work needed for each algorithm should also be considered. From a practical point of view, the local Lipschitz constants L_{ij} will need to be determined as the algorithm proceeds, and in all likelyhood this can only be done approximately. While (2.2) requires only a determination of the Lipschitz constant L, (2.4) requires L_{ij} to be determined for each subrectangle that is to be tested. Thus, an algorithm based on (2.4) will certainly require more work than one based on (2.2).

Consider the important special case of $F(x) = \|f(x)\|_\infty$, where $f : \mathscr{R} \to \mathbb{R}^d$ is a C^1 vector field on a neighborhood of $\mathscr{R}$. Let $x, y \in \mathscr{R}$. Integrating the derivative of $f(y + s(x - y))$ from

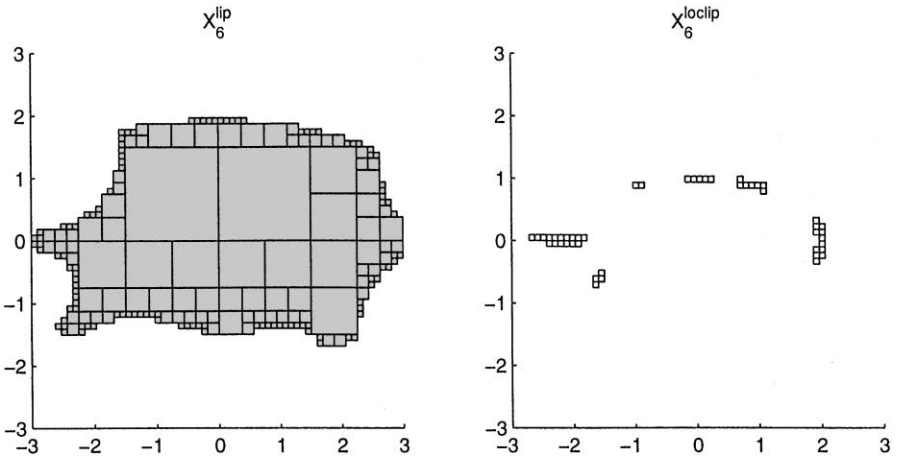

Fig. 2. The sets X_6^{lip} and X_6^{loclip} are the unions of the shaded rectangles in the respective plots. The computation of X_6^{lip} required 3728 work units and the computation of X_6^{loclip} required 13 568 work units.

$s = 0$ to $s = 1$ gives

$$f(x) - f(y) = \int_0^1 Df(y + s(x - y))(x - y)\,\mathrm{d}s, \tag{2.5}$$

where $Df(x)$ denotes the derivative matrix of the vector field $f(x)$. From the definition of the induced matrix norm ($\|A\|_\infty = \max\{\|Ax\|_\infty : \|x\|_\infty = 1\}$), and the properties of the vector norm and integral, it follows that $|F(x) - F(y)| \leqslant L\|x - y\|_\infty$, for all $x, y \in \mathscr{R}$, where $L = \max\{\|Df(x)\|_\infty : x \in \mathscr{R}\}$. Similarly, $L_{ij} = \max\{\|Df(x)\|_\infty : x \in \mathscr{R}_{ij}\}$ can be used in (2.4). These values can be estimated (from below) by sampling $\|Df(x)\|_\infty$ at points in $\mathscr{R}_{ij}$.

In this context, to measure the amount of work done while using one of these subdivision algorithms we will use function evaluations. One evaluation of $f(x)$ will be called one work unit. By analogy, the evaluation of $Df(x)$ requires d work units. In the next example, we compare the algorithms based on (2.2) and (2.4), both in terms of rate of convergence and in terms of the number of work units used.

Example 2.2. Consider the vector field $f(x, y) = [f_1, f_2]$ defined on the plane $\mathbb{R}^2$ by

$$f_1(x, y) = x^2 + 4y^2 - 4, \quad f_2(x, y) = y(x - 1.995)(y - x^2)(y - x + 1).$$

This is a simple polynomial vector field having eight zeros, all of which lie on the ellipse $f_1 = 0$. Although the zeros are all isolated, there are three zeros clustered together in a small neighborhood of the point $(2, 0)$. To three decimal places they are $(2, 0)$ and $(1.995, \pm 0.071)$. To compare algorithms (2.2) and (2.4) they were both started with $\mathscr{R} = [-3, 3] \times [-3, 3]$, $F(x) = \|f(x)\|_\infty$ and allowed to process through six levels of recursion, as described in Example 2.1. The resulting sets X_6^{lip} and X_6^{loclip} are shown in Fig. 2. As suggested above, sampling was used to estimate the Lipschitz constants. On larger rectangles 16 evaluations of $Df(x)$ were used for these estimates, and on smaller rectangles only one evaluation of $Df(x)$ was used. Gradations between these two extremes were also used. The

total work done using (2.2) was 3728 work units and the total work done using (2.4) was 13,568 work units.

There are many other choices of selection criteria. As our last example, we consider an alternative to (2.4) that is suggested by changing the norm used in defining F. Consider $F(x)=\|f(x)\|_2$, where f is C^1 on a neighborhood of $\mathscr{R}=(a_1,b_1)\times\cdots\times(a_d,b_d)$. As before let x_{ij} denote the mid-point of R_{ij}. Since matrix multiplication can be written as $Ax=x_1A_1+\cdots+x_dA_d$, where A_j is the jth column of A and x_j is the jth component of x, it follows from (2.5) that $|F(x)-F(x_{ij})|\leqslant\tau_{ij}$, for all $x\in R_{ij}$, where

$$\tau_{ij}=\frac{1}{2}\sum_{k=1}^{d}\left(\max_{x\in R_{ij}}\left\|\frac{\partial f}{\partial x_k}(x)\right\|_2\right)|b_k-a_k|. \tag{2.6}$$

The numbers τ_{ij} defined in this way are directionally sensitive to the rates of change in F. These numbers determine an algorithm based on the following selection criterion. Let $X_0^\tau=\mathscr{R}$ and, for each $i\geqslant 1$, inductively define

$$J_i^\tau=\{j\in\{1,2,\ldots,n_i\}\colon R_{ij}\subset X_{i-1}^\tau \text{ and } F(x_{ij})\leqslant 2^{-i}+\tau_{ij}\} \tag{2.7}$$

and the corresponding subsets $X_i^\tau=\bigcup\{R_{ij}\colon j\in J_i^\tau\}$. As in the other cases this is a nested sequence that converges to Z (see Section 6).

3. Known zeros — isolation and exclusion

The subdivision algorithms described in the previous section can be compared to the bisection algorithm. As such when the location of a zero has been sufficiently resolved, it becomes desirable to accelerate the convergence if possible. For example, if $F(x)=\|f(x)\|$, where $f(x)$ is a C^1 vector field on a neighborhood of the rectangle $\mathscr{R}$, one could attempt to solve $f(x)=0$ using Newton's method given this situation. However, even if this is successful, to accomplish our goal of determining $Z=\{x\in\mathscr{R}\colon F(x)=0\}$ there is still the problem of finding all the remaining zeros in $\mathscr{R}$. In this section an approach to this problem, that does not effect the recursive geometry of the subdivision process, is presented. The basic idea is to use a type of penalty function that we choose to call an exclusion function. We develop the necessary ingredients for an algorithmic process in two parts. In Section 4, these ingredients will be incorporated into a recursive process that can be used to find all the zeros of $f(x)$.

As in the previous section, $\mathscr{R}\subset\mathbb{R}^d$ will denote a rectangle with sides parallel to the coordinate planes, and $\mathscr{P}=\{\mathscr{P}_i\}_{i\geqslant 0}$, with $\mathscr{P}_i=\{R_{ij}\}_{j=1}^{n_i}$, the sequence of partitions of $\mathscr{R}$ into congruent subrectangles. The midpoint and radius of R_{ij} will be denoted by x_{ij} and ρ_i, respectively.

3.1. Subdivision with exclusion

Let $F:\mathscr{R}\to[0,\infty)$ be Lipschitz continuous, with constant L, and suppose that $x_0\in\mathscr{R}$ is a known isolated zero of $F(x)$. In this context isolated means there is a relative deleted neighborhood $N(x_0,\delta)=\{x\in\mathscr{R}\colon 0<\|x-x_0\|_\infty<\delta\}$ of x_0 in $\mathscr{R}$ such that $F(x)>0$ for all $x\in N(x_0,\delta)$. Such a neighborhood will be called an isolating neighborhood. The l_∞ norm is used due to the rectangular

geometry of $\mathscr{R}$ and its children. Since x_0 is known it should be excluded from the set of zeros in $\mathscr{R}$ that remain to be found. A type of penalty function can be used for this purpose.

Let $\delta > 0$ be a fixed diameter for an isolating neighborhood $N(x_0, \delta)$ of x_0. Suppose that $\phi : [0, \infty) \to [0, \infty)$ is a continuous function satisfying: (i) $\phi(t)$ is non-increasing, (ii) $\phi(t) > 0$ for $0 \leqslant t < \delta$ and (iii) $\phi(t) = 0$ for $t > \delta$. An "*exclusion function*" $e(x)$ is then defined by $e(x) = \phi(\|x - x_0\|_\infty)$. This terminology is justified by the following fact verified in Section 6. If we apply the algorithm based on (2.1) to $F_e(x) = F(x) + e(x)$ then $X_\infty = \{x \in \mathscr{R}: F_e(x) = 0\} = \{x \in \mathscr{R} - \{x_0\}: F(x) = 0\}$.

A subdivsion with exclusion selection criterion that is analogous to the one based on (2.2) can now be developed. In doing so, a device is used that eliminates the need to actually evaluate $e(x)$ when the algorithm is implemented. Instead, a simple test for the inclusion $R_{ij} \subset N(x_0, \delta)$ can be used. Let $\gamma \in (0, \delta)$, $\beta > 0$, and define $\phi(t)$ to be the continuous function that is linear on the subintervals $[0, \delta - \gamma]$, $[\delta - \gamma, \delta]$ and satisfies $\phi(0) = \phi(\delta - \gamma) = \beta$, $\phi(t) = 0$ for $t > \delta$. Also define $\chi(t)$ to be the characteristic function for the interval $[0, \delta - \gamma]$: $\chi(t) = 1$, if $t \in [0, \delta - \gamma]$, and $\chi(t) = 0$ otherwise. To enforce the exclusion feature of the algorithm we use the function

$$\eta(t, s) = \phi(t) - \beta\chi(t + s), \quad t, s \geqslant 0, \tag{3.1}$$

to define the numbers $\eta_{ij} = \eta(\|x_{ij} - x_0\|, \rho_i)$. Let $X_0^{\text{elip}} = \mathscr{R}$ and for each $i \geqslant 1$ inductively define

$$J_i^{\text{elip}} = \{1 \leqslant j \leqslant n_i: R_{ij} \subset X_{i-1}^{\text{elip}} \quad \text{and} \quad F(x_{ij}) + e(x_{ij}) \leqslant 2^{-i} + \rho_i L + \eta_{ij}\}, \tag{3.2}$$

and the corresponding subset $X_i^{\text{elip}} = \bigcup\{R_{ij}: j \in J_i^{\text{elip}}\}$. Again it can be shown (see Section 6) that $X_\infty^{\text{elip}} = \{x \in \mathscr{R} - \{x_0\}: F(x) = 0\}$.

The inequality in the definition of J_i^{elip} can be equivalently written as

$$F(x_{ij}) \leqslant 2^{-i} + \rho_i L - \beta\chi(\|x_{ij} - x_0\| + \rho_i).$$

When $\|x_{ij} - x_0\| + \rho_i > \delta - \gamma$ this inequality reduces to the one appearing in (2.2). When $\|x_{ij} - x_0\| + \rho_i \leqslant \delta - \gamma$, so that R_{ij} is contained in the closure of a $\delta - \gamma$ neighborhood of x_0, the inequality becomes

$$F(x_{ij}) \leqslant 2^{-i} + \rho_i L - \beta.$$

This will never be valid if β is chosen sufficiently large. Since β is arbitrary this can be assumed to be the case. Hence, either $\|x_{ij} - x_0\| + \rho_i \leqslant \delta - \gamma$ and R_{ij} should be discarded, or the reverse inequality is true and the inequality to be tested reduces to the one in (2.2). Thus, evaluations of $e(x)$ and $\eta(t, s)$ are never actually needed to determine J_i^{elip}, and $F(x_{ij})$ only needs to be computed when $\|x_{ij} - x_0\| + \rho_i > \delta - \gamma$. Obviously, a local Lipschitz version of subdivision with exclusion can also be given.

3.2. Estimating isolating neighborhoods

To implement the subdivision with exclusion algorithm based on the selection criterion (3.2) an isolating neighborhood must be known. That is, a radius δ which determines an isolating neighborhood must be computed. We consider this problem for $F(x) = \|f(x)\|_2$, where $f(x)$ is C^2 vector field defined on a neighborhood of $\mathscr{R}$ and $x_0 \in \mathscr{R}$ is a zero of f. If the derivative matrix $Df(x_0)$ is nonsingular then x_0 is an isolated zero according to the inverse function theorem. For convenience set $A = Df(x_0)$, and define the remainder term $r(x, x_0) = f(x) - f(x_0) - Df(x_0)(x - x_0) = f(x) - A(x - x_0)$.

Since A is invertible it follows that $A^{\mathrm{T}}A$ is a symmetric positive definite matrix. Hence, it has a smallest positive eigenvalue $\lambda_1 > 0$ such that $\|Ax\|_2^2 \geqslant \lambda_1 \|x\|_2^2$, for every $x \in \mathbb{R}^d$.

Consider the family of relative-deleted neighborhoods, $N(x_0, s) = \{x \in \mathscr{R}: 0 < \|x - x_0\|_\infty < s\}$, for $s > 0$. Since x_0 is isolated, $N(x_0, s)$ will be an isolating neighborhood for s sufficiently small. Define the function

$$b(s) = \sup_{x \in N(x_0, s)} \frac{\|r(x, x_0)\|_2}{\|x - x_0\|_\infty^2}, \quad s > 0. \tag{3.3}$$

Since $f \in C^2$ the function $b(s)$ is a finitely valued, nondecreasing function whose range is contained in $[0, B]$, for some $B > 0$ which depends only on $\mathscr{R}$. If $x \in N(x_0, s)$ it follows from the triangle inequality and the definitions of $\lambda_1, b(s)$ that

$$\begin{aligned} \|f(x)\|_2 &\geqslant \|A(x - x_0)\|_2 - \|r(x, x_0)\|_2 \\ &\geqslant \sqrt{\lambda_1}\|x - x_0\|_2 - b(s)\|x - x_0\|_\infty^2 \geqslant \sqrt{\lambda_1}\|x - x_0\|_\infty - b(s)\|x - x_0\|_\infty^2. \end{aligned}$$

If $\|x - x_0\|_\infty < \delta(s)$, where $\delta(s) = \sqrt{\lambda_1}/b(s)$ then the lower bound on the right is positive. Hence, $F(x) = \|f(x)\|_2 > 0$, for all $x \in N(x_0, s) \cap N(x_0, \delta(s))$. Clearly, $N(x_0, s) \cap N(x_0, \delta(s)) = N(x_0, \min\{s, \delta(s)\})$. Since $b(s)$ is nondecreasing, $\delta(s)$ is nonincreasing. It follows that there is a unique $s_0 > 0$ such that $s_0 = \delta(s_0) = \max\{\min\{s, \delta(s)\}: s > 0\}$. Thus, with s_0 defined in this way, $F(x) = \|f(x)\|_2 > 0$, for $x \in N(x_0, s_0)$.

These observations yield an algorithm for computing a relatively large isolating neighborhood of x_0 in $\mathscr{R}$. Suppose s is chosen to be the radius ρ of the rectangle $\mathscr{R}$. Assuming λ_1 has been computed, compute $b(\rho)$ and then $\delta(\rho)$ to determine a radius $\min\{\rho, \delta(\rho)\}$ of an isolating neighborhood. An examination of the graph of $\min\{s, \delta(s)\}$ on $s > 0$ shows that either $\rho \leqslant \delta(\rho)$, so $\mathscr{R}$ is an isolating neighborhood of x_0, or $\delta(\rho) < \rho$, in which case $[\delta(\rho), \rho)$ is an interval estimate of the optimal radius s_0. In the later case, a new value of s can be chosen, based on the interval estimate, and a larger radius computed.

4. Accelerating convergence

A main goal of this work is to describe an effective numerical algorithm that can be used to find all the isolated zeros of a vector field $f(x)$ lying in a given rectangle $\mathscr{R}$. A subdivision algorithm, such as one of those described in Section 2, provides the main ingredient. As with any iterative process the basic issues of efficiency, rates of convergence and stopping criteria must be addressed.

Although the various subdivision algorithms discussed in Section 2 have convergence properties analogous to the bisection method, some converge faster than others, in the sense that they produce smaller sets approximating the set of zeros at the same depth of recursion. This is clearly illustrated in Fig. 2. However, as Example 2 also shows increasing the rate of convergence may decrease efficiency. Clearly, the difference in the rate of convergence is due to having better information on the behavior of the function on the current rectangle being tested. In fact, the selection methods given in (2.2) and (2.4) are at two extremes, the former using only the global information L, which incurs no cost, and the later using the most local information available, which incurs a considerable cost. To balance the competition between the rate of convergence and efficiency we have thought to use an intermediate strategy of updating the Lipschitz constants intermittently.

Consider for example a rectangle in the plane. After proceeding to a recursion depth of 2 the original rectangle has been subdivided into 16 subrectangles. At this point, a local Lipschitz constant can be computed for each of the subrectangles that has been retained and then used subsequently for all of the subrectangle's respective children and grandchildren. The choice of 2 here is an arbitrary one and is not being suggested as optimal.

As with the bisection method, once a reasonably good approximation of a zero has been determined a more rapidly converging method can be used to accelerate convergence. In the context of a subdivision algorithm, a good approximation would take the form of a small cluster of subrectangles that are isolated from the rest of the subrectangles that have been retained at a certain depth of recursion. Intuitively, a subdivision algorithm should yield a collection of small clusters of retained subrectangles, each clustering around a zero of $F(x)$. This behavior is evident in the more resolved graph of Fig. 2. Some issues arise in considering these clusters of rectangles. Assuming that a zero can be found, for example by Newton's method, how does the cluster get subsequently processed? Since there may be other zeros contained in the cluster of subrectangles the cluster cannot be automatically discarded. From a data-management perspective, how should the subrectangles in the cluster be stored? The depth of recursion needed to produce isolated clusters may have also produced a large number a very small rectangles. One strategy that can be used to overcome both of these difficulties is the use of covering rectangles.

Suppose that C is the union of a cluster of subrectangles. We say R is a covering rectangle for C if R is a rectangle, with sides parallel to the coordinate planes, that contains C. Algorithmically, we are interested in the covering rectangle that is minimal with respect to this property. Replacing a cluster C by the minimal covering rectangle R completely resolves the data-management problem. Furthermore since $C \subset R$ any zeros that remain in C can be found by applying a subdivision with exclusion algorithm to R. This yields a second recursive process which starts with a recursive subdivision process being applied to a root rectangle $\mathscr{R}$. Clusters of subrectangles are generated and then incorporated into covering rectangles. Each covering rectangle, having the same characteristics as the root rectangle, then becomes a root rectangle for a new process.

Covering rectangles can be constructed as the algorithm proceeds without the need to save any of the individual retained subrectangles. With the maximum depth of subdivision recursion fixed, the subdivision algorithm generates retained subrectangles one at a time when it reaches the maximum depth. If a retained subrectangle does not intersect any of the existing covering rectangles then it becomes its own covering rectangle, otherwise the subrectangle is incorporated into any covering rectangle that it intersects, perhaps causing covering rectangles to be enlarged or merged as part of the process.

The use of covering rectangles and subdivision with exclusion can be combined into a composite algorithm which is recursive but can be expected to terminate after a finite number of steps, with the set of zeros Z completely determined. We describe such an algorithm as a procedure (or subroutine) that takes three input parameters $\mathscr{R}, \mathscr{S}, \mathscr{D}$, where (i) $\mathscr{R}$ is a rectangle with sides parallel to the coordinate planes, (ii) $\mathscr{S}$ is either the empty set, $\emptyset$, or a two element set, $\{x_0, \delta\}$, where x_0 is an isolated zero of $f(x)$ in $\mathscr{R}$ and $N(x_0, \delta)$ is an isolating neighborhood of x_0, and (iii) D is a positive integer indicating the maximum depth of recursion allowed for the subdivision process on $\mathscr{R}$.

To allow for a generic description we assume that a selection criterion has been chosen (including a strategy for updating any Lipschitz-like constants) and use $\{J_i\}_{i=0}^{D}$ to denote the sequence of subsets of indices that are obtained by applying the selection criterion to $\mathscr{R}$ and then recursively

to its offspring to a recursion depth of D. If $\mathscr{S} \neq \emptyset$ then it is assumed that a subdivsion with exclusion algorithm is used and the corresponding sequence of subsets of indices will be denoted by $\{J_i^e\}_{i=0}^D$. Let the corresponding subsets of retained subrectangles be denoted by $X_i = \bigcup\{R_{ij}: j \in J_i\}$, or $X_i^e = \bigcup\{R_{ij}: j \in J_i^e\}$, respectively, for $1 \leqslant i \leqslant D$. Using these notations we define a recursive composite subdivision algorithm as follows.

Input: $\mathscr{R}, \mathscr{S}, D$.

Step 1: If $\mathscr{S} = \emptyset$, then recursively apply the subdivision algorithm to depth D to determine X_D, otherwise use the data in $\mathscr{S}$ to define an exclusion function and use subdivsion with exclusion to determine X_D^e. In either case call the resulting set X_D^*.

Step 2: If $X_D^* = \emptyset$, the procedure terminates; there are no zeros in $\mathscr{R}$ (resp. $\mathscr{R} - \{x_0\}$). Otherwise, X_D^* has a finite number of connected components $C_1, \ldots, C_M$. For $m = 1, \ldots, M$, let $\mathscr{R}_m$ be the smallest rectangle, with sides parallel to coordinate planes, that encloses C_m; that is let $\mathscr{R}_m$ be the minimal covering rectangle for C_m. If $\mathscr{R}_1, \ldots, \mathscr{R}_M$ is a disjoint set of covering rectangles then this step is complete. Otherwise, consolidate intersecting covering rectangles into larger covering rectangles until all covering rectangles are disjoint. In either case, let $\mathscr{R}_1, \ldots, \mathscr{R}_M$ denote the disjoint set of covering rectangles obtained at the end of this step.

Step 3: For each covering rectangle $\mathscr{R}_m$, $1 \leqslant m \leqslant M$, obtained at the end of Step 2, attempt to find one zero in $\mathscr{R}_m$, for example by using Newton's method. If a zero x_m can be found, use the observations of Section 3.2 to determine a radius $\delta_m > 0$ for an isolating neighborhood $N(x_m, \delta_m)$ of x_m and set $\mathscr{S}_m = \{x_m, \delta_m\}$. Otherwise set $\mathscr{S}_m = \emptyset$.

Step 4: For $m = 1, \ldots, M$, perform precisely one of the following three tasks: (i) If $\mathscr{S}_m = \emptyset$ then recursively pass the parameters $\mathscr{R}_m, \emptyset, D$ to this procedure. (ii) If $\mathscr{S}_m \neq \emptyset$ and $\mathscr{R}_m$ is contained in $N(x_m, \delta_m)$ then add x_m to the list of zeros. (iii) If $\mathscr{S}_m \neq \emptyset$ and $\mathscr{R}_m$ is not contained in $N(x_m, \delta_m)$ then add x_m to the list of zeros and recursively pass the parameters $\mathscr{R}_m, \mathscr{S}_m, D$ to this procedure.

Output: $Z = \{x \in \mathscr{R}: f(x) = 0\}$.

In the above algorithm the recursion will continue as long as one of the covering rectangles obtained in Step 2 may contain an undetermined zero. This is the condition being tested in Step 4. The algorithm will terminate either at the beginning of Step 2, if all subrectangles have been discarded during the subdivision process of Step 1, or at the end of Step 4, if each covering rectangle $\mathscr{R}_m$ assembled in Step 2 satisfies $\{x_m\} \subset \mathscr{R}_m \subset N(x_m, \delta_m)$, where $\mathscr{S}_m = \{x_m, \delta_m\}$ is the corresponding set determined in Step 3. From a practical point of view, since the algorithm is recursive, an upper bound on the number of generations of covering rectangles recursively spawned needs to be set. Thus, in a practical implementation the output may also include covering rectangles that were spawned at the maximum depth of recursion and thus remain unresolved subrectangles. There is another possibility for algorithm failure. The subdivision process may yield clusters of subrectangles for which the original rectangle is the minimal covering rectangle.

5. Zeros of vector fields — some examples

The application of a composite subdivision algorithm to several example problems is discussed in the section. The algorithms described in the previous sections were implemented as programs

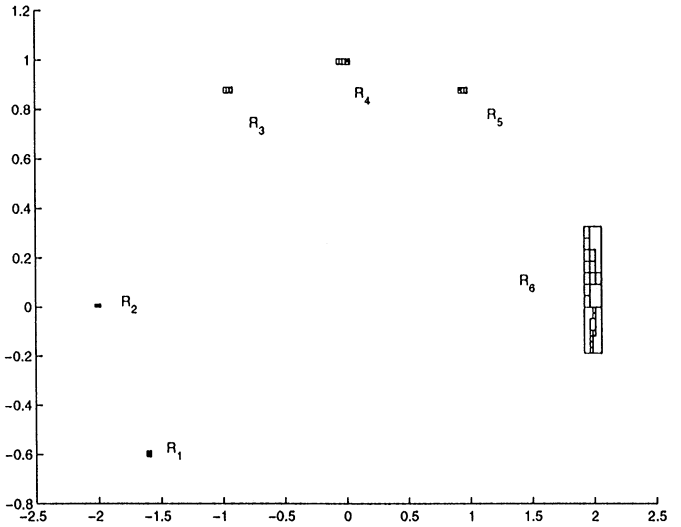

Fig. 3. The covering rectangles $\mathscr{R}_1,\ldots,\mathscr{R}_6$ obtained at the end of Step 2 during the first pass through the composite algorithm, starting with $\mathscr{R} = [-3,3] \times [-3,3]$. The function is $F(x,y) = \|f(x,y)\|_2$, where f is the vector field in Example 2.2. The set X_8 is the union of the shaded rectangles contained in all of the covering rectangles.

written in the *C* programming language. The basic objects of the algorithm, rectangles, were declared as structures with members describing attributes such as vertices, Lipschitz constants, and the recursion level to which the rectangle belonged. The addressing scheme described in the appendix was used in determining subrectangles of a given rectangle. Since *C* functions can be called recursively, both the subdivision process and the composite algorithm were readily implemented as *C* functions. Other functions were written to manage the covering rectangles, implement Newton's method, compute radii for isolating neighborhoods, and estimate Lipschitz constants. The selection criterion used was essentially the one described in (2.7), but with a strategy for updating the τ_{ij} that was intermediate to a purely local one (e.g., Eq. (2.4)) and a purely global one (e.g., Eq. (2.2)). All of the computations were performed on an AlphaStation 255/300 workstation, running a Digital Unix V4.0D operating system. The source code for the programs can be obtained by contacting the first author (mwsmiley@iastate.edu) or downloaded from the web (http://www.public.iastate.edu/~mwsmiley/zeros.html). For convenience, below we refer to our implementation of the recursive composite subdivision algorithm described in the previous section as RCSA.

5.1. Examples

Example 5.1. As a first example, we consider the vector field $f(x,y) = [f_1, f_2]$ defined in Example 2.2, but with $F(x,y) = \|f(x,y)\|_2$, and describe the step by step workings of the RCSA procedure for this problem. The procedure was started with the parameters $\mathscr{R} = [-3,3] \times [-3,3]$, $\mathscr{S} = \emptyset$ and $D = 8$. At the end of Step 2, six covering rectangles $\mathscr{R}_1,\ldots,\mathscr{R}_6$ are obtained. These are shown in Fig. 3, enclosing the set of subrectangles that make up X_8. Fig. 4 contains blow-ups of each of the six covering rectangles. In Step 3, a zero is found in each of the covering rectangles by using Newton's with the midpoint of the covering rectangle as the initial approximation. Radii

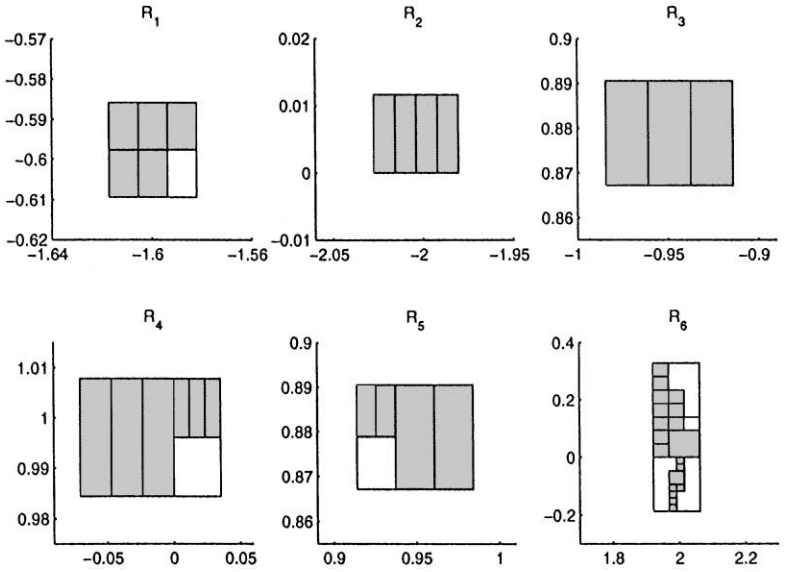

Fig. 4. Blow-ups of the covering rectangles $\mathscr{R}_1,\ldots,\mathscr{R}_6$ appearing in Fig. 3. The shaded rectangles in each case make up the connected components of X_8.

for isolating neighborhoods are then computed and Step 3 finishes with $\mathscr{S}_m=\{x_m,\delta_m\}$, $1\leqslant m\leqslant 6$. In Step 4, $\mathscr{R}_1,\ldots,\mathscr{R}_5$ are found to be contained in the computed isolating neighborhoods of their respective zeros, while $\mathscr{R}_6$ is not. (This is as it should be since $\mathscr{R}_6$ contains the cluster of three zeros near $(2,0)$ and hence there are two other zeros that remain to be determined.) Thus, in Step 4 six zeros are added to the list of zeros and there is one recursive call to the procedure with $\mathscr{R}_6,\mathscr{S}_6,D$ passed as parameters.

Starting with the input $\mathscr{R}_6,\mathscr{S}_6,D$ causes subdivision with exclusion to be used in Step 1 since $\mathscr{S}_6$ is nonempty. At the end of Step 2, four covering rectangles $\mathscr{R}_{61},\ldots,\mathscr{R}_{64}\subset\mathscr{R}_6$ are found to cover the clusters of saved subrectangles that result. Newton's method produces a zero in two of the covering rectangles and fails to converge in the other two. Thus, Step 3 concludes with $\mathscr{S}_{61}=\{x_{61},\delta_{61}\}$, $\mathscr{S}_{62}=\{x_{62},\delta_{62}\}$, $\mathscr{S}_{63}=\emptyset$, $\mathscr{S}_{64}=\emptyset$. In Step 4, $\mathscr{R}_{61}$ and $\mathscr{R}_{62}$ are found to be contained in the computed isolating neighborhoods of their respective zeros. Hence, the two zeros x_{61},x_{62} are added to the list of zeros and two recursive calls to the procedure are made. Both calls, one with the parameters $\mathscr{R}_{63},\emptyset,D$ and one with the parameters $\mathscr{R}_{64},\emptyset,D$ yield $X_D=\emptyset$ at the end of the Step 1. Thus, after an initial call and two recursive calls to the procedure the recursion is terminated with all eight zeros found. To complete its task our implementation of the composite algorithm used approximately 7.3×10^3 works units, which is considerably less than the number used to compute X_6^{loclip} as described in Example 2.2. Varying the choice of D caused variations in the number of work units but this count was quite typical.

Example 5.2. A vector field $f(x,y,z)=[f_1,f_2,f_3]$ with known solutions is defined on $\mathbb{R}^3$ by

$$f_1=(x^2+y^2+z^2-r_1^2)(x^2+y^2+z^2-r_2^2),$$

$$f_2 = (a_0x + b_0y + c_0z - d_{01})(a_0x + b_0y + c_0z - d_{02}),$$

$$f_3 = \prod_{i=1}^{m}(a_ix + b_iy + c_iz - d_i),$$

where (assuming $c_0 \neq 0$)

$$a_i = b_0v_i - c_0\sin\theta_i, \quad d_i = d_{01}c_i/c_0,$$

$$b_i = c_0\cos\theta_i - a_0v_i, \quad v_i = -(a_0\cos\theta_i + b_0\sin\theta_i)/c_0,$$

$$c_i = a_0\sin\theta_i - b_0\cos\theta_i, \quad \theta_i = \mathrm{i}\pi/m.$$

With this choice of coefficients the level surface $f_3 = 0$ defines a collection of m planes that contain a common line $\mathscr{L}$ that is perpendicular to the two parallel planes $f_2 = 0$. The angle π/m is the angle between each pair of adjacent planes. Thus, the intersection of the two level surfaces $f_2 = 0$ and $f_3 = 0$ is a set of $2m$ lines. It follows that there are $8m$ solutions of $f(x, y, z) = 0$ when the parameters are chosen so that each line intersects both spheres $f_1 = 0$ (i.e. when the common line $\mathscr{L}$ intersects the planes $f_2 = 0$ at points interior to the smaller sphere). This happens for example when $r_1 = 1, r_2 = 2$, $a_0 = 0.5$, $b_0 = 0.5$, $c_0 = 1$, $d_{01} = 0.2$, and $d_{02} = -0.7$. The RCSA algorithm was applied in this case, with $m = 3$. The procedure was started with $\mathscr{R} = [-3, 3] \times [-3, 3] \times [-3, 3]$, $\mathscr{S} = \emptyset$ and $D = 8$. All 24 zeros were determined and are listed in Table 1. The number of work units used was approximately 7.5×10^5. Thus, the code took about the same amount of time to execute as it did to compile. The algorithm was also applied with larger values of m and with different values for the coefficients. Not surprisingly, as m increased so did the amount of work. For example, with $m = 5$ about 1.8×10^6 work units were needed, and with $m = 7$ about 13.5×10^6 work units were needed. In both cases all solutions (40 and 56, respectively) were found.

Example 5.3. The method of alternative problems can be used (cf. [13]) to show that all solutions of the boundary value problem

$$-(x^2u')' + u^3 - \lambda u = f(x), \quad 1 < x < 4, \tag{5.1}$$

$$u(1) = u(4) = 0. \tag{5.2}$$

can be characterized in terms of the zeros of a vector field $B(\omega)$ defined on $\mathscr{R}^d$, where d depends on λ. The zeros ω are in fact the first d Fourier coefficients of the corresponding solution of (5.1), (5.2). The vector field $B(\omega)$ is called the bifurcation function and is defined in terms of integrals involving solutions of an auxiliary boundary value problem. Having an effective way to find all solutions of the equation $B(\omega) = 0$, and hence also of (5.1), (5.2), was our main purpose in attempting to develop a robust zero finding method. We have found the RCSA algorithm to be a very effective tool in attacking this problem. Computed solutions and bifurcation diagrams can be found in (cf. [13]).

Example 5.4. The polynomial system

$$0 = a_{11}x_1^4 + a_{12}x_1^3x_2 + a_{13}x_1^3 + a_{14}x_1 + a_{15},$$

$$0 = a_{21}x_1x_2^2 + a_{22}x_2^2 + a_{23}$$

Table 1
The 24 zeros of the function $f(x)$ defined in Example 5.2, with $r_1 = 1$, $r_2 = 2$, $a_0 = 0.5$, $b_0 = 0.5$, $c_0 = 1$, $d_{01} = 0.2$, $d_{02} = -0.7$, and $m = 3$

i	x_i	y_i	z_i
1	−1.933009	−0.300000	0.416504
2	−1.701684	0.000000	1.050842
3	−1.258803	1.360696	−0.750946
4	−1.044691	−1.589843	0.617267
5	−0.996600	1.726162	−0.164780
6	−0.951026	−0.300000	−0.074486
7	−0.800000	0.000000	0.600000
8	−0.776373	−1.344718	1.260546
9	−0.717665	0.423418	−0.552876
10	−0.592072	−0.805884	−0.001021
11	−0.499927	0.865900	0.017013
12	−0.360640	−0.624646	0.692643
13	0.082249	−0.962075	−0.260086
14	0.085220	0.367221	−0.926221
15	0.453788	0.785984	−0.419886
16	0.464511	−0.804557	0.370022
17	0.511026	−0.300000	−0.805513
18	0.623386	1.151180	−1.544510
19	0.869521	−1.899353	−0.062016
20	0.869521	1.506056	−0.987788
21	0.960000	0.000000	−0.280000
22	0.961183	−1.664819	0.551817
23	1.493009	−0.300000	−1.296504
24	1.861684	0.000000	−0.730842

Table 2
Coefficients for the polynomial systems of Example 5.4

i	a_{i1}	a_{i2}	a_{i3}	a_{i4}	a_{i5}
1	1.069$e-03$	2.000e + 04	1.000e + 00	−1.800e − 10	−1.283e − 24
2	2.000e + 16	1.000e + 14	−1.000e + 00		
i	c_{i1}	c_{i2}	c_{i3}	c_{i4}	c_{i5}
1	1.069e − 05	2.000e + 02	1.000e + 05	−1.800e + 05	−1.283e − 04
2	2.000e − 02	1.000e + 01	−1.000e + 01		

with the coefficients given in Table 2 arises in the study of chemical equilibria (cf. [9,10]). It is known to have seven real solutions and a complex conjugate pair of solutions. These solutions are very ill-conditioned and appear at extreme scales. Thus, scaling of the variables and equations is usually required for solution techniques to be effective (see [9] for scaling strategies). As an example, changing variables $x_1 = t_1 \times 10^{-5}$, $x_2 = t_2 \times 10^{-7}$ and scaling the first equation by 10^{20} and the second

Table 3
Computed solutions for Example 5.4

i	t_{i1}	t_{i2}
1	−1.34298	−1.00134
2	−1.34030	1.00134
3	1.34030	0.99866
4	1.34298	−0.99866

by 10 results in the system

$$0 = c_{11}t_1^4 + c_{12}t_1^3t_2 + c_{13}t_1^3 + c_{14}t_1 + c_{15},$$

$$0 = c_{21}t_1t_2^2 + c_{22}t_2^2 + c_{23}$$

with the coefficients given in Table 2. The RCSA procedure was applied to this scaled system. Starting the procedure with $\mathscr{R} = [-5,5] \times [-5,5]$, $\mathscr{S} = \emptyset$ and $D = 9$, four of the seven real solutions were found. The process used approximately 3×10^4 work units. The computed solutions, after rounding to 6 digits, are given in Table 3 and are in close agreement with the corresponding solutions given in [9]. In terms of the original variables, the other three real-valued solutions are very roughly $(x_1,x_2) = (-7 \times 10^{-15}, \pm 1 \times 10^{-7})$ and $(x_1,x_2) = (-5 \times 10^{-3}, -5 \times 10^{-5})$. Clearly, these require a different scaling. Of all the solutions, only the one with both x_1 and x_2 positive is physically relevant.

Example 5.5. In [6], Hsu and Guttalu considered the problem of finding all the period 2 points for the mapping $g: \mathbb{R}^4 \to \mathbb{R}^4$ defined by

$$g_1(x) = x_1 + C_1[x_3 - \alpha \sin x_1 \cos x_2], \quad g_3(x) = D_1[x_3 - \alpha \sin x_1 \cos x_2],$$

$$g_2(x) = x_2 + C_2[x_4 - \alpha \cos x_1 \sin x_2], \quad g_4(x) = D_2[x_4 - \alpha \cos x_1 \sin x_2].$$

This is a problem that has its origins in mechanics (see [5] for a formulation of the problem). The constants C_1, C_2, D_1, D_2 that appear are defined in terms of two parameters μ_1, μ_2 according to $D_1 = \exp(-2\mu_1)$, $D_2 = \exp(-2\mu_2)$, $C_1 = (1 - D_1)/2\mu_1$, $C_2 = (1 - D_2)/2\mu_2$. As reported in [6], with $\alpha = 5$, $\mu_1 = \pi/10$ and $\mu_2 = \pi/5$ there are 41 period 2 points in the rectangle $\mathscr{R} = [-1.02\pi, 1.02\pi] \times [-1.02\pi, 1.02\pi] \times [-0.5\pi, 0.5\pi] \times [-0.5\pi, 0.5\pi]$. Clearly, the problem of finding period 2 points of $g(x)$ is equivalent to finding zeros of $f(x) = g(g(x)) - x$. We applied the RCSA algorithm to this problem, starting with the rectangle just described. The algorithm computed exactly 41 zeros of $f(x)$ that were in complete agreement with the period 2 points listed in [6]. The amount of work units was roughly 4×10^9 and execution took almost 40 min.

Example 5.6. The kinematics of a general six-degree-of-freedom manipulator can be described by a system of the form $H_j(\theta_1, \theta_2, \theta_4, \theta_5) = 0$, $1 \leqslant j \leqslant 4$, where

$$H_j(\theta) = a_{1,j} \cos\theta_1 \cos\theta_2 + a_{2,j} \cos\theta_1 \sin\theta_2 + a_{3,j} \sin\theta_1 \cos\theta_2 + a_{4,j} \sin\theta_1 \sin\theta_2$$

$$+ a_{5,j} \cos\theta_4 \cos\theta_5 + a_{6,j} \cos\theta_4 \sin\theta_5 + a_{7,j} \sin\theta_4 \cos\theta_5 + a_{8,j} \sin\theta_4 \sin\theta_5$$

Table 4
Coefficient values for the functions H_j of Example 5.6

i	$a_{i,1}$	$a_{i,2}$	$a_{i,3}$	$a_{i,4}$
1	−2.90965281e − 02	3.40782577e − 02	−6.02977989e − 01	4.78568871e − 01
2	1.23862738e − 01	−1.56062187e − 01	−1.31668277e − 01	1.12420352e − 01
3	2.15085388e − 02	−2.70999144e − 02	−7.58247387e − 01	6.47403003e − 01
4	1.67560227e − 01	−1.96248865e − 01	1.04706029e − 01	−8.31026124e − 02
5	0.00000000e + 00	2.20738619e − 01	−5.51846548e − 02	3.90625000e − 02
6	−7.00449588e − 02	0.00000000e + 00	1.23100969e − 01	1.75112397e − 02
7	−2.70632939e − 01	0.00000000e + 00	3.18608753e − 02	6.76582347e − 02
8	0.00000000e + 00	−8.52868532e − 01	2.13217133e − 01	−1.01101189e − 02
9	−6.15842911e − 01	7.21283769e − 01	−2.14660298e − 02	1.96624145e − 04
10	4.55239233e − 01	−5.73583561e − 01	−6.01805218e − 01	5.00438377e − 01
11	−1.30935803e − 01	−6.31988451e − 02	0.00000000e + 00	−5.00000000e − 01
12	−1.29409523e − 01	0.00000000e + 00	2.44181587e − 01	5.05897097e − 01
13	4.18258152e − 01	−1.45259532e − 01	3.63148829e − 02	−2.64395380e − 02
14	−5.41265877e − 01	0.00000000e + 00	−2.09664074e − 02	1.95686833e − 01
15	0.00000000e + 00	−4.75625621e − 01	−7.13438432e − 01	1.95312500e − 01
16	1.50925910e − 01	0.00000000e + 00	6.15504846e − 01	2.26388866e − 01
17	−2.38536450e − 02	1.91169833e − 02	5.47700901e − 01	−3.39187451e − 01

$$+ a_{9,j}\cos\theta_1 + a_{10,j}\sin\theta_1 + a_{11,j}\cos\theta_2 + a_{12,j}\sin\theta_2 + a_{13,j}\cos\theta_4 + a_{14,j}\sin\theta_4$$

$$+ a_{15,j}\cos\theta_5 + a_{16,j}\sin\theta_5 + a_{17,j}.$$

There are two additional angles θ_3 and θ_6 that can be determined from $\theta_1, \theta_2, \theta_4, \theta_5$. Here we have conformed to the notation used by Tsai and Morgan [16]. By changing variables ($x_1 = \cos\theta_1$, $x_2 = \sin\theta_2$, etc.) and adding four new equations ($x_1^2 + x_2^2 = 1$, etc.) this can be converted to a polynomial system. This is done in [16] so that homotopy continuation methods for polynomial systems can be applied. This problem is also discussed in [10]. The RCSA algorithm can be applied directly to this system. This was done using the coefficients given in Table 4. This is a different set of coefficients than given in [10]. The coefficients given in Table 4 correspond to the data given in [16], and were used so that our computed solutions could be checked against the values of $\theta_1, \theta_2, \theta_4, \theta_5$ recorded in [16].

The system of equations $H_j(\theta_1, \theta_2, \theta_4, \theta_5)=0$, $1 \leqslant j \leqslant 4$, is obtained in such a way that extraneous solutions are introduced. According to [16] there are 16 genuine solutions and 16 extraneous solutions. Of the 16 genuine solutions, 12 are real-valued and physically relevant while four are complex-valued and not physically relevant. Starting the RCSA algorithm with the rectangle $\mathscr{R} = [-\pi, \pi]^4$ we were able to compute all 12 physically relevant solutions. These are recorded in Table 5. However, only seven of the extraneous solutions were found. Since the algorithm returned several unresolved covering rectangles upon reaching the pre-set upper bound on recursive generations of covering rectangles, we are led to believe that the remaining real-valued extraneous solutions are tightly packed in a small region where the function is nearly singular. For example, with the generation limit set to six there were three unresolved covering rectangles returned, all of which were contained in the rectangle $[2.842, 3.052] \times [1.411, 1.519] \times [1.031, 1.227] \times [-1.602, -1.472]$. (coordinates given in rad). Restarting the RCSA procedure with this rectangle resulted in essentially the same algorithm

Table 5
The 12 physically relevant solutions of the system $H_j(\theta_1, \theta_2, \theta_4, \theta_5) = 0$, $1 \leqslant j \leqslant 4$, of Example 5.6. The angles θ_i are given in degrees

i	$\theta_{i,1}$	$\theta_{i,2}$	$\theta_{i,4}$	$\theta_{i,5}$
1	−142.999715	100.072114	18.464582	−59.490607
2	−106.069054	−140.856892	−161.281104	35.539996
3	−65.365854	142.240676	−70.901651	−51.633556
4	−16.694202	97.897535	−80.984287	−25.722033
5	7.747473	103.865780	−21.369854	−79.895876
6	20.933357	58.740169	−27.073033	−125.660752
7	38.928126	−56.446153	12.283461	72.225890
8	47.258567	163.443114	28.317628	−41.132867
9	107.559134	1.998782	166.772114	−173.540089
10	115.859496	−168.646343	157.169857	−111.407314
11	120.516644	31.270039	114.146527	−143.618716
12	167.676727	83.550094	65.842958	−88.668795

behavior; slightly smaller unresolved covering rectangles were returned. Complex-valued solutions were not sought. The number of work units in this case was roughly 1×10^9 and execution took approximately 20 min.

5.2. Further remarks

The results that we have obtained illustrate both the strengths and weaknesses of our zero finding subdivision algorithm. They show the algorithm can be effective in handling physically relevant nontrivial problems which have many zeros. They also show the dependence of the computational work on the dimension of the problem and on the complexity of the set of zeros. As with the bisection method, the RCSA method appears to have robust convergence properties and requires only basic information of the system function. However, it can be slow to converge, especially as the dimension grows.

In developing programs for the example problems, we experimented with several strategies for implementing selection criteria, approximating Lipschitz constants and performing other required tasks. Experimentation showed that subtle variations in strategies could produce substantial changes in algorithm performance, as measured in terms of work units. For example, by delaying the creation of covering rectangles, at the expense of additional memory requirements, we were able to reduce the work unit count in Example 5.5 by a factor of 3. Thus, we believe the code we developed should be considered as a preliminary version. Additional efforts to optimize performance are needed and could result in substantial improvements in performance.

6. Convergence of the algorithms

The convergence of the subdivision algorithms described in the previous sections relies only on the basic properties of continuity and compactness. Thus, throughout this section we assume that

(X,d) is a compact metric space and $F:X\to[0,+\infty)$ is a continuous function. The set of zeros of F in X will be denoted by Z; hence, $Z=F^{-1}(0)\cap X=\{x\in X\colon F(x)=0\}$. In this setting successive subdivisions of X are described in terms of a sequence of partitions $\{\mathscr{P}_i\}_{i=0}^{\infty}$ of X. This sequence is assumed to satisfy the two hypotheses:

($\mathscr{H}_1$) Each partition, $\mathscr{P}_i=\{K_{ij}\}_{j=1}^{n_i}$, is composed of a finite collection of compact subsets. Thus,

$$X=\bigcup_{j=1}^{n_i}K_{ij}\quad\forall i\geqslant 0,$$

where each K_{ij} is a compact subset of X and n_i is a finite integer. The partition $\mathscr{P}_0$ consists only of X, so that $n_0=1$ and $K_{01}=X$

($\mathscr{H}_2$) For each $i\geqslant 1$, the partition $\mathscr{P}_i$ is a sub-partition of $\mathscr{P}_{i-1}$ which has been further refined. That is, for each $j\in\{1,2,\ldots,n_i\}$ there is a $j'\in\{1,2,\ldots,n_{i-1}\}$ such that $K_{ij}\subset K_{i-1,j'}$. The fineness of the partition is characterized by the number

$$d_i=\max_{1\leqslant j\leqslant n_i}\operatorname{diam}(K_{ij}),\quad\text{where }\operatorname{diam}(K_{ij})=\sup_{x,y\in K_{ij}}d(x,y).$$

It is assumed that $d_{i+1}\leqslant d_i$ and $d_i\to 0$ as $i\to\infty$.

Since all of the subsets are to be compact the void intersection property of a partition, $K_{ij_1}\cap K_{ij_2}=\emptyset$ when $j_1\neq j_2$, will not be required. Properties ($\mathscr{H}_1$) and ($\mathscr{H}_2$) are obviously valid when X is a rectangle in $\mathscr{R}^d$ partitioned by subdivisions into congruent subrectangles as described in Section 2. The following lemma provides a basic tool for verifying convergence of all the subdivision algorithms discussed in this paper.

Lemma 6.1. *Let $\{\varepsilon_i\}_{i=1}^{\infty}$ be a decreasing sequence of positive numbers, with $\varepsilon_i\to 0$ as $i\to\infty$. Let $X_0=X$ and, for each $i\geqslant 1$, define the subset of indices*

$$J_i=\left\{j\in\{1,2,\ldots,n_i\}\colon\min_{x\in K_{ij}}F(x)\leqslant\varepsilon_i\right\}\tag{6.1}$$

and the corresponding subset $X_i=\bigcup\{K_{ij}\colon j\in J_i\}$. Set $X_\infty=\bigcap\{X_i\colon i\geqslant 0\}$. Then $Z=X_\infty$.

Proof. If $Z=\emptyset$ then there is a point $x_0\in X$ such that $F(x)\geqslant F(x_0)>0$, for all $x\in X$. Hence, $X_i=\emptyset$ for all i sufficiently large and the equality is valid. Therefore, Z may be assumed nonempty. In this case, each X_i is also nonempty since $Z\subset X_i$. To see this let $x\in Z$. Then, for each $i\geqslant 0$, there is a $j\in\{1,2,\ldots,n_i\}$ such that $x\in K_{ij}$ and

$$0=F(x)=\min_{y\in K_{ij}}F(y)\leqslant\varepsilon_i.$$

Hence, $j\in J_i$ and $x\in X_i$.

It is now clear that X_∞ is a nonempty set that contains Z. To see that it is also contained in Z let $x\in X_\infty$. Since $x\in X_i$ for each i there is a sequence of points $\{y_i\}_{i\geqslant 1}$ satisfying $y_i\in X_i$, $d(x,y_i)\leqslant d_i$ and $F(y_i)\leqslant\varepsilon_i$, for each $i\geqslant 1$. This is clear since $x\in K_{ij}$, for some $j\in J_i$, with $\operatorname{diam}(K_{ij})\leqslant d_i$ by properties ($\mathscr{H}_1$), ($\mathscr{H}_2$), and by compactness there is a $y_i\in K_{ij}$ such that

$$F(y_i)=\min_{y\in K_{ij}}F(y)\leqslant\varepsilon_i.$$

Since $F(x) \leqslant |F(x) - F(y_i)| + F(y_i)$, it follows by passing to the limit as $i \to \infty$ that $F(x) = 0$. Thus, $X_\infty \subset Z$ and the equality $Z = X_\infty$ follows. □

Corollary 6.1. *Let (Y, d_Y) be a metric space, $y_0 \in Y$ be a fixed element of Y, and $f : X \to Y$ a continuous function. If $F(x) = d_Y(f(x), y_0)$ and J_i, X_i are defined as in Lemma* 6.1, *then $\{x \in X : f(x) = y_0\} = \bigcap\{X_i : i \geqslant 0\}$.*

We now suppose that $F(x)$ is Lipschitz continuous on X, with Lipschitz constant L. And further assume that, as part of the construction of the sequence of partitions $\{\mathscr{P}_i\}_{i=0}^{\infty}$, a sample point $x_{ij} \in K_{ij}$ is selected (e.g., the midpoint of a rectangle). The distance to the sampling point from any point $x \in K_{ij}$ must be no larger than

$$d_{ij} = \max_{x \in K_{ij}} d(x, x_{ij}).$$

Obviously, $d_{ij} \leqslant d_i$ for all $j \in \{1, 2, \ldots, n_i\}$, and $|F(x) - F(x_{ij})| \leqslant d_{ij} L$ for all $x \in K_{ij}$. To allow for a slightly more general situation we assume that for each $i \geqslant 1$, $1 \leqslant j \leqslant n_i$, there is a number τ_{ij} such that

$$|F(x) - F(x_{ij})| \leqslant \tau_{ij} \quad \forall x \in K_{ij}, \tag{6.2}$$

$$\tau_{ij} \leqslant C d_i, \quad 1 \leqslant j \leqslant n_i \tag{6.3}$$

for some constant C that is independent of i, j. Clearly, $\tau_{ij} = d_{ij} L$ is one example. Two other examples with $K_{ij} = R_{ij}$ a subrectangle and x_{ij} its midpoint are: (i) $\tau_{ij} = \rho_i L_{ij}$, where L_{ij} is a local Lipschitz constant as given in (2.3) and $\rho_i = d_{ij}$ is the radius of R_{ij}; and (ii) the numbers τ_{ij} defined in (2.6).

Lemma 6.2. *Let $\{\varepsilon_i\}_{i=1}^{\infty}$ be a decreasing sequence of positive numbers, with $\varepsilon_i \to 0$ as $i \to \infty$, and let $\{\tau_{ij}\}$ be a set of numbers satisfying* (6.2), (6.3). *Set $X_0^\tau = X$ and, for each $i \geqslant 1$, inductively define the subset of indices*

$$J_i^\tau = \{j \in \{1, 2, \ldots, n_i\} : K_{ij} \subset X_{i-1}^\tau \text{ and } F(x_{ij}) \leqslant \varepsilon_i + \tau_{ij}\}$$

and the corresponding subset $X_i^\tau = \bigcup\{K_{ij} : j \in J_i^\tau\}$. Set $X_\infty^\tau = \bigcap\{X_i^\tau : i \geqslant 0\}$. Then $Z = X_\infty^\tau$.

Proof. Again it may be assumed that Z is nonempty. We assume the induction hypothesis that $X_{i-1} \subset X_{i-1}^\tau$. Let J_i be defined by (6.1), or equivalently (see the discussion following (2.1))

$$J_i = \left\{ j \in \{1, 2, \ldots, n_i\} : K_{ij} \subset X_{i-1} \text{ and } \min_{x \in K_{ij}} F(x) \leqslant \varepsilon_i \right\}.$$

Let $j \in J_i$ and $x \in K_{ij}$ be such that $F(x) \leqslant \varepsilon_i$. From (6.2) it follows that

$$F(x_{ij}) \leqslant \varepsilon_i + |F(x_{ij}) - F(x)| \leqslant \varepsilon_i + \tau_{ij}.$$

Hence, $j \in J_i^\tau$ which shows $J_i \subset J_i^\tau$ and consequently $X_i \subset X_i^\tau$. It now follows from Lemma 6.1 that $Z = X_\infty \subset X_\infty^\tau$. Let $x \in X_\infty^\tau$. For each $i \geqslant 1$ there is an index $j \in J_i^\tau$ such that $x \in K_{ij}$. By (6.2), (6.3)

$$F(x) \leqslant \varepsilon_i + \tau_{ij} + |F(x) - F(x_{ij})| \leqslant \varepsilon_i + 2Cd_i.$$

Letting $i \to \infty$ then shows that $x \in Z$. Thus, $X_\infty^\tau \subset Z$. □

Next, we verify the convergence of the subdivision with exclusion algorithms presented in Section 3, under the assumptions on $\phi(x)$ and $e(x)$ postulated there.

Lemma 6.3. *Let $\{\varepsilon_i\}_{i=1}^{\infty}$ be a decreasing sequence of positive numbers, with $\varepsilon_i \to 0$ as $i \to \infty$. Set $X_0^e = X$, and for each $i \geqslant 1$, inductively define the subset of indices*

$$J_i^e = \left\{ j \in \{1,2,\ldots,n_i\}\colon \min_{x \in K_{ij}} \{F(x) + e(x)\} \leqslant \varepsilon_i \right\}, \tag{6.4}$$

and the corresponding subset $X_i^e = \bigcup\{K_{ij}\colon j \in J_i^e\}$. Set $X_\infty^e = \bigcap\{X_i^e\colon i \geqslant 0\}$. Then $Z - \{x_0\} = X_\infty^e$.

Proof. Applying Lemma 6.1 to $F_e(x) = F(x) + e(x)$ shows that X_∞^e is the set of all zeros of $F(x) + e(x)$. Since both terms must be zero the conclusion follows. □

In addition to the function $\eta(t,s)$ defined in (3.1), the function $\zeta(t,s) = \phi(t) - \phi(t+s)$ will also be needed in proving the convergence of the algorithm based on the selection criterion (3.2). With x_{ij} and d_{ij} defined as above set $\eta_{ij} = \eta(\|x_{ij} - x_0\|, d_{ij})$.

Lemma 6.4. *Let $\{\varepsilon_i\}_{i=1}^{\infty}$ be a decreasing sequence of positive numbers, with $\varepsilon_i \to 0$ as $i \to \infty$, and let $\{\tau_{ij}\}$ be a set of numbers satisfying* (6.2), (6.3). *Set $X_0^{e\tau} = X$, and for each $i \geqslant 1$, inductively define the subset of indices*

$$J_i^{e\tau} = \{j \in \{1,2,\ldots,n_i\}\colon K_{ij} \subset X_{i-1}^{e\tau} \text{ and } F(x_{ij}) + e(x_{ij}) \leqslant \varepsilon_i + \tau_{ij} + \eta_{ij}\},$$

and the corresponding subset $X_i^{e\tau} = \bigcup\{K_{ij}\colon j \in J_i^{e\tau}\}$. Set $X_\infty^{e\tau} = \bigcap\{X_i^{e\tau}\colon i \geqslant 0\}$. Then $Z - \{x_0\} = X_\infty^{e\tau}$.

Proof. We assume the induction hypothesis that $X_{i-1}^e \subset X_{i-1}^{e\tau}$. Let J_i^e be defined by (6.4), or equivalently

$$J_i^e = \left\{ j \in \{1,2,\ldots,n_i\}\colon K_{ij} \subset X_{i-1}^e \text{ and } \min_{x \in K_{ij}} \{F(x) + e(x)\} \leqslant \varepsilon_i \right\}.$$

Let $j \in J_i^e$ and choose $x \in K_{ij}$ satisfying $F(x) + e(x) \leqslant \varepsilon_i$. Since

$$\|x - x_0\| \leqslant \|x - x_{ij}\| + \|x_{ij} - x_0\| \leqslant d_{ij} + \|x_{ij} - x_0\|$$

and $\phi(t)$ is nonincreasing with $\phi(t) \geqslant \beta\chi(t)$ for all t, it follows that

$$\begin{aligned} e(x_{ij}) - e(x) &= \phi(\|x_{ij} - x_0\|) - \phi(\|x - x_0\|) \\ &\leqslant \phi(\|x_{ij} - x_0\|) - \phi(\|x_{ij} - x_0\| + d_{ij}) \\ &\leqslant \phi(\|x_{ij} - x_0\|) - \beta\chi(\|x_{ij} - x_0\| + d_{ij}) = \eta_{ij}. \end{aligned}$$

Note that similarly $e(x) - e(x_{ij}) \leqslant \zeta(\|x - x_0\|, d_{ij})$. From the first of these inequalities it follows that

$$F(x_{ij}) + e(x_{ij}) \leqslant \varepsilon_i + |F(x_{ij}) - F(x)| + e(x_{ij}) - e(x) \leqslant \varepsilon_i + \tau_{ij} + \eta_{ij}.$$

Thus, $j \in J_i^{e\tau}$, $J_i^e \subset J_i^{e\tau}$ and $Z - \{x_0\} = X_\infty^e \subset X_\infty^{e\tau}$. Suppose that $x \in X_\infty^{e\tau}$. Then for each $i \geqslant 1$ there is a $j_i \in J_i^{e\tau}$ such that $x \in K_{ij_i}$. But then

$$\begin{aligned} F(x) + e(x) &\leqslant F(x_{ij_i}) + e(x_{ij_i}) + |F(x) - F(x_{ij_i})| + e(x) - e(x_{ij_i}) \\ &\leqslant \varepsilon_i + 2Cd_i + \eta_{ij_i} + \zeta(\|x - x_0\|, d_{ij_i}). \end{aligned}$$

Since $d_i \to 0$ and $x_{ij_i} \to x$ as $i \to \infty$, passing to the limit as $i \to \infty$ in this inequality produces

$$F(x) + e(x) \leqslant \eta(\|x - x_0\|, 0) = \phi(\|x - x_0\|) - \beta\chi(\|x - x_0\|).$$

But $e(x) = \phi(\|x - x_0\|)$. Thus, the validity of this equality implies that $F(x) = 0$ and $\|x - x_0\| > \delta - \gamma$. Thus $X_\infty^{e\tau} \subset \{x \in \mathscr{R} - \{x_0\}: F(x) = 0\}$. □

Appendix

There is a simple indexing scheme that can be used to conveniently list all the subrectangles obtained through subdivision of a given rectangle. Let $\mathscr{R} \subset \mathbb{R}^d$ be a rectangle with sides parallel to the coordinate planes. Such a rectangle is determined by a pair of opposing vertices $a, b \in \mathbb{R}^d$ according to

$$\mathscr{R} = \prod_{i=1}^{d} [a_i, b_i] = \{x \in \mathbb{R}^d: a_i \leqslant x_i \leqslant b_i,\ 1 \leqslant i \leqslant d\}.$$

A partition of $\mathscr{R}$ into congruent rectangles can be obtained as follows. Let $N \geqslant 2$ be a positive integer determining the number of subdivisions in each coordinate direction. For example, if $N = 2$ and $\mathscr{R}$ is a rectangle in the plane, then each side will be subdivided into two subintervals and there will be four subrectangles obtained by taking all pairwise products of subintervals. We may refer to these subrectangles as children of the parent rectangle $\mathscr{R}$. In general, there will be $n_c = N^d$ children (subrectangles) in the partition, which will be enumerated as $\{R_0, \ldots, R_{n_c - 1}\}$.

The vertices of the subrectangles $\{R_i\}_{i=0}^{n_c - 1}$ are conveniently listed by using a multi-index notation. Let $\alpha = (\alpha_1, \ldots, \alpha_d)$ be a multi-index with the property that each $\alpha_j \in \{0, 1, \ldots, N - 1\}$. Then α naturally corresponds to an integer expressed in base N:

$$\alpha \leftrightarrow i = \sum_{k=0}^{d-1} \alpha_{d-k} N^k.$$

For example, if $N = 2$ then each α corresponds to a binary number which determines an integer in the range from 0 to $2^d - 1$. In general, for each $i \in \{0, \ldots, n_c - 1\}$ there is a unique multi-index $\alpha^i = (\alpha_1^i, \ldots, \alpha_d^i)$ such that $\alpha_1^i \alpha_2^i \ldots \alpha_d^i$ is the base N representation of i. Using this correspondence, the vertices of the child R_i are readily described in terms of the vertices a, b of the parent rectangle $\mathscr{R}$. Let $v = N^{-1}(b - a) \in \mathbb{R}^d$ and define the componentwise product $\alpha^i * v = (\alpha_1^i v_1, \ldots, \alpha_d^i v_d) \in \mathbb{R}^d$. Then R_i is the rectangle with opposing vertices $c_i, d_i \in \mathbb{R}^d$ given by

$$c_i = a + \alpha^i * v \quad \text{and} \quad d_i = c_i + v.$$

References

[1] E.L. Allgower, K. Georg, Continuation and path following, Acta Numer. (1993) 1–64.
[2] M. Dellnitz, A. Hohmann, A subdivision algorithm for the computation of unstable manifolds and global attractors, Numer. Math. 75 (1997) 293–317.
[3] N. Dyn, The work of John Gregory, rational spline interpolation, subdivision algorithms and C^2 polygonal patches, in: G. Mullineux (Ed.), The Mathematics of Surfaces, Vol. VI, Clarendon Press, Oxford, 1996, pp. 21–42.

[4] J.A. Gregory, An introduction to bivariate uniform subdivision, Numerical Analysis 1991, in: D.F. Griffiths, G.A. Watson (Eds.), Pitman Research Notes in Mathematics, Longman, New York, 1991, pp. 103–117.
[5] C.S. Hsu, Nonlinear behavior of multibody systems under implusive parametric excitation, in: K. Magnus (Ed.), Dynamics of Multibody Systems, Springer, New York, 1978, pp. 63–74.
[6] C.S. Hsu, 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.
[7] C.S. Hsu, W.H. Zhu, A simplical mapping method for locating the zeros of a function, Quart. Appl. Math. 42 (1984) 41–59.
[8] B. Kearfott, An efficient degree-computation method for a generalized method of bisection, Numer. Math. 32 (1979) 109–127.
[9] K. Meintjes, A.P. Morgan, A methodology for solving chemical equilibrim systems, Appl. Math. Comput. 22 (1987) 333–361.
[10] A.P. Morgan, A.J. Sommese, L.T. Watson, Finding all isolated solutions to polynomial systems using HOMPACK, ACM Trans. Math. Software 15 (1989) 93–122.
[11] V.Y. Pan, Solving a polynomial equation: some history and recent progress, SIAM Rev. 39 (1997) 187–220.
[12] W.C. Rheinboldt, Numerical Analysis of Parametrized Nonlinear Equations, Wiley, New York, NY, 1986.
[13] M.W. Smiley, C. Chun, Computation of Morse decompositions for semilinear elliptic PDEs, Numer. Methods for PDEs, to be published.
[14] M. Sosonkina, L.T. Watson, D.E. Stewart, A note on the end game in homotopy zero curve tracking, ACM Trans. Math. Software 22 (1996) 281–287.
[15] F. Stenger, Computing the topological degree of a mapping in $\mathbb{R}^n$, Numer. Math. 25 (1975) 23–38.
[16] L.-W. Tsai, A.P. Morgan, Solving the kinematics of the most general six- and five-degree-of-freedom manipulators by continuation methods, J. Mech. Transmissions Automat. Des. 107 (1985) 189–200.
[17] M.N. Vrahatis, K.L. Iordanidis, A rapid generalized method of bisection for solving systems of non-linear equations, Numer. Math. 49 (1986) 123–138.
[18] M.N. Vrahatis, Solving systems of nonlinear equations using the nonzero value of the topological degree, ACM Trans. Math. Software 14 (1988) 312–328.
[19] L.T. Watson, Globally convergent homotopy methods: A tutorial, Appl. Math. Comput. 31 (1989) 369–396.
[20] H. Weyl, Randbermerkungen zu hauptproblemen der mathematik, II, fundamentalsatz der algebra and grundlagen der mathematik, Math. Z. 20 (1924) 131–151.