THE ENCLOSURE OF SOLUTIONS OF PARAMETER-DEPENDENT SYSTEMS OF EQUATIONS

by A. Neumaier

Universität Freiburg, D-7800 Freiburg, West Germany

1. Introduction

In this paper we shall consider a covering method for the enclosure of the solution set of a finite-dimensional system of nonlinear equations

(1.1) $F(\tilde{x}) = 0,$

where F is a function defined on a subset $D \subseteq \mathbb{R}^n$ with values in $\mathbb{R}^m$, $m \leq n$. If F is regular in D, i.e. if f is continuously differentiable and $F'(\tilde{x})$ has rank m for all $\tilde{x}$ in a neighbourhood of the solution set

$$M = \{\tilde{x} \in D \mid F(\tilde{x}) = 0\},$$

then the solution set M of (1) is a p-dimensional manifold in $\mathbb{R}^n$, $p = n-m$. If F is regular then, in the degenerate case $p = 0$ (i.e. $n = m$), the equation (1.1) has only finitely many solutions. In case $p > 0$, the vector $\tilde{x}$ of variables often contains p distinguished variables $\lambda_1, \ldots, \lambda_p$, called parameters, and the solution or solutions of (1) are sought in dependence on $\lambda_1, \ldots, \lambda_p$, corresponding to a parametrization of M in terms of $\lambda_1, \ldots, \lambda_p$. In that case it may be more convenient to separate the independent variables and the parameters, and write (1.1) as

(1.2) $F(\tilde{x}, \tilde{\lambda}) = 0,$

where now $F: D \subseteq \mathbb{R}^m \times \mathbb{R}^p \to \mathbb{R}^m$.

The standard method for the solution of (1.1) (or (1.2)) is the continuation method, developed primarily for the case $p = 1$. Here, in the regular case, the solution set M consists of a union of disjoint curves, and the continuation method essentially consists in starting at a particular solution $\tilde{x}^0$ and approximately following the solution curve containing $\tilde{x}^0$ in one or both of the two possible directions. A neighbouring solution point is obtained by approximately linearizing (1.1) and solving the resulting equation, while taking care that the linearization error remains small. Certain variations which use

piecewise-linear approximations of (1) are also known as simplicial methods. The case p>1 is reduced to p=1 by tracing particular solution curves on M, and the case p=0 is treated by embedding the solution of (1.1) into a 1-dimensional manifold by introducing an artificial parameter (homotopy or fixed point methods). The details of the method change from author to author. See e.g. Schwetlick [22], Zangwill and Garcia [25], Rheinboldt [20] and the survey paper [21] of Rheinboldt in these proceedings.

Clearly, to find all solutions of (1.1) with a continuation method, one needs to know at least one point on each connected component of M. While in some cases this is easy to achieve, in others it is more difficult, and in certain problems of practical interest even the number of components of M is unknown (Ushida and Chua [24]). For the unreliability of homotopy methods (and related deflation methods) for finding all solutions in case p=0, cf. Allgower and Georg [2] (in particular Example II).

Interval methods for the solution of (1.1) are based on a different, global approach. As is natural in applications, it is assumed that only that part of M is interesting for which all variables lie within certain bounds,

$$l_i \le \tilde{x}_i \le u_i \quad (i = 1,\ldots,n),$$

so that only the solutions of (1) contained in a box $x^0=[l,u]\in\mathbb{IR}^n$ are sought. Let us write, for any $x\in\mathbb{ID}$ (the set of interval boxes contained in D),

$$\Sigma(F,x) := \{\tilde{x}\in x \mid F(\tilde{x})=0\}$$

The covering method to be discussed in this paper consists in covering the set $\Sigma(F,x^0)$ by a collection of smaller and smaller boxes which give increasingly accurate information about the location of the solution set. At the most refined stage, the covering computed for the solution set describes the solution set as accurately as required for representing it as an image on a computer screen. And indeed, I was inspired to the presented investigations partly by recent applications of interval arithmetic to computer-aided geometric design (CAGD); see Mudur and Koparkar [7,13], Toth [23].

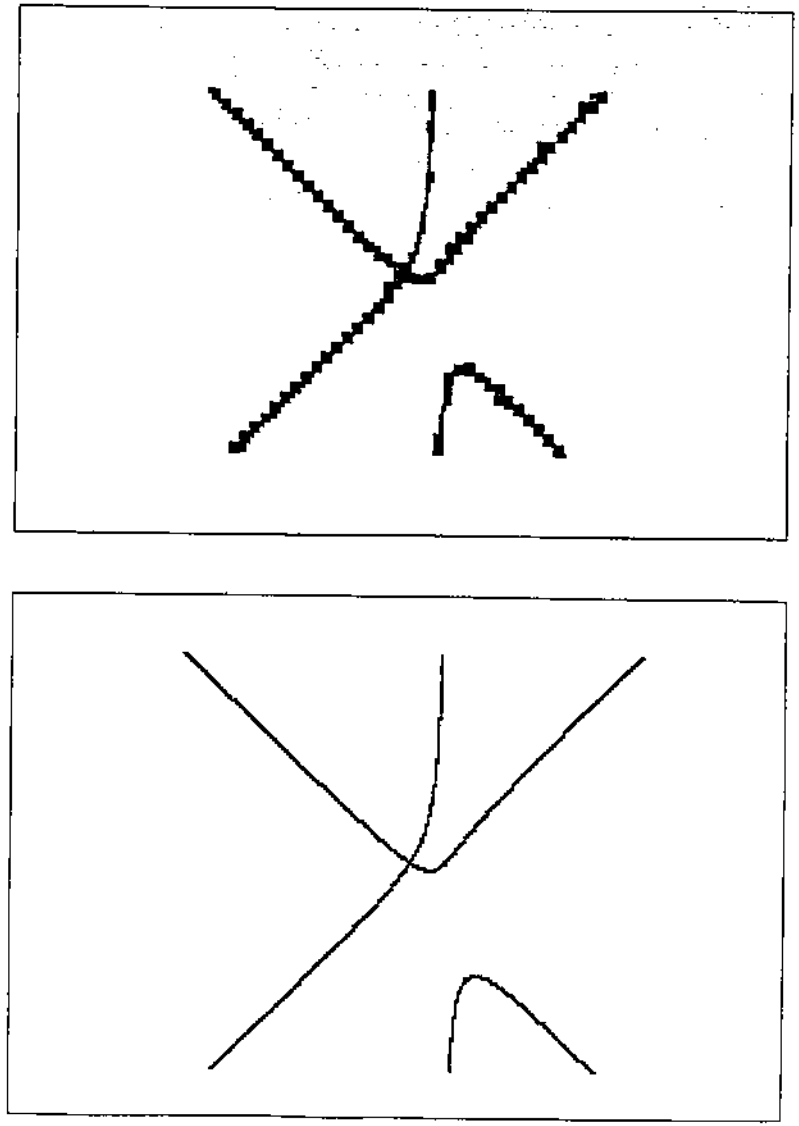


Fig. 1: A coarse and a fine covering for the curve $x^3-xy^2+x^2-xy-y^2=0$ (see Example 3 in §4).

Clearly, the covering method needs no information about particular solutions or even the number of solution curves. On the other hand, it requires a criterion for testing whether a box $x \in \mathbb{IR}^n$ contains no solution of (1.1) (i.e. $\Sigma(F,x)=\emptyset$), and thus can be discarded from the covering set. The simplest test uses an interval extension of F, i.e. a mapping from $D' \subseteq \mathbb{IR}^n \to \mathbb{IR}^m$ (also denoted by F) such that

$$\tilde{x} \in x \Rightarrow F(\tilde{x}) \in F(x);$$

clearly

$$0 \notin F(x) \Rightarrow \Sigma(F,x)=\emptyset.$$

If the interval extension is also continuous, so that in particular $F(x) \to F(\tilde{x})$ if $x \to \tilde{x}$, then, for sufficiently small boxes, this test excludes any point not on the solution set. Since there are many ways to construct continuous interval extensions for functions defined by arithmetical expressions (Moore [12]), this shows the feasibility of the approach. However, in this form the method is very slow (cf. Morgan and Shapiro [11]); hence more sophisticated tests are needed to speed up the process and to yield narrow enclures without using too many boxes. We discuss the covering method in more detail in Section 2. In Section 3 we give some details about the solution of linear interval equations, required for a more refined test for discarding irrelevant parts of a box. Sample results for a particular realization of the covering method are presented in Section 4.

The terminology used in this paper follows Neumaier [14,15]. In particular, $\mathbb{IR}$ and $\mathbb{IR}^n$ denote the set of real closed intervals, and of interval vectors of dimension n, respectively. $\underline{x}$, $\bar{x}$, $\check{x}$, and $\varrho(x)$ denote the lower bound, upper bound, midpoint, and radius of $x \in \mathbb{IR}^n$, int x is the interior of $x \in \mathbb{IR}^n$, and $\square S := [\inf S, \sup S]$ (with $\square S = \emptyset$ if $S = \emptyset$) denotes the interval hull of a bounded set $S \subseteq \mathbb{IR}^n$. Thin intervals (interval vectors) are those with $\underline{x} = \bar{x}$. Furthermore, we shall need the ternary operator Γ, defined for $a,b,x \in \mathbb{IR}$ by

$$\Gamma(a,b,x) = \square\{\tilde{x} \in x \mid \tilde{a}\tilde{x}=\tilde{b} \text{ for some } \tilde{a} \in a,\ \tilde{b} \in b\}. \tag{1.3}$$

$\Gamma(a,b,x)$ is either an interval, or the empty set, and it can be computed from the endpoints of a,b,x as

$$\Gamma(a,b,x) = \begin{cases} b/a \cap x & \text{if } 0 \notin a, \\ \square(x \setminus \operatorname{int}[\underline{b}/\underline{a}, \underline{b}/\bar{a}]) & \text{if } b>0 \in a, \\ \square(x \setminus \operatorname{int}[\bar{b}/\bar{a}, \bar{b}/\underline{a}]) & \text{if } b<0 \in a, \\ x & \text{if } 0 \in a,b. \end{cases} \tag{1.4}$$

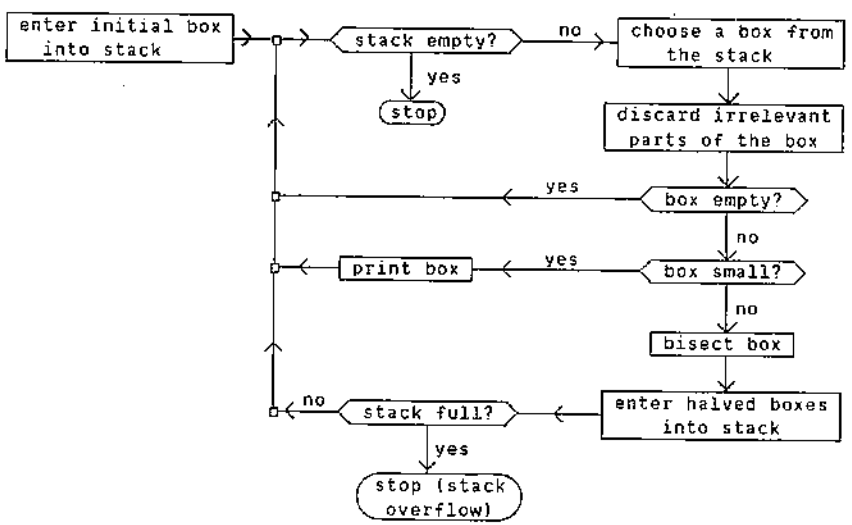
Apart from some compares, the calculation of $\Gamma(a,b,x)$ requires at most one floating-point division if $x \geq 0$ or $x \leq 0$, and at most two otherwise.

2. Covering the solution set

In this section we discuss the following diagram describing the various ingredients of an algorithm for adaptively covering the solution set $\Sigma(F,x^0)$ of the bound-constrained nonlinear system

$$F(\tilde{x}) = 0, \quad \tilde{x} \in x^0, \tag{2.1}$$

where $F: D \subseteq \mathbb{R}^n \to \mathbb{R}^m$, $m \leq n$, and $x^0 \in \mathbb{I}D$.



Remarks concerning the practical implementation are given with reference to the programming language PASCAL-SC (see [1,4]), which was also used to compute the numerical results in Section 4.

(i) The stack. Since the order in which the boxes are processed is immaterial, the list of boxes can be simply stored in a last in-first out stack. This is simply an array *stack* of boxes, together with an integer *last* which remembers the last entry of *stack* used at the present stage.

```
const nov = {number of variables}; bmax = {stack size};
type box   = array[1..nov] of interval;
var  stack: array[1..bmax] of box;
```

A box x is entered into the stack by

```
last := last+1; if last ≤ bmax then stack[last] := x;
```

and chosen from the stack by

```
if last > 0 then x := stack[last]; last := last-1;
```

The stack is empty if *last* = 0 and full if *last* > *bmax*; if this happens the algorithm stops. An empty stack indicates the successful completion of the algorithm; a full stack indicates failure due to lack of storage (bmax too small) or unreasonably high accuracy requirements (test for smallness too strict).

(ii) Discarding irrelevant parts of the box. This is the heart of the algorithm. Let us assume that F is given by an arithmetical expression and that the interval evaluation $F(x^0)$ of F at the interval box is defined. Then we can compute F(x) for all boxes $x \subseteq x^0$, using the interval arithmetic provided e.g. by PASCAL-SC. If $0 \notin F(x)$ then x contains no solution point and is discarded (i.e. replaced by $\emptyset$). Since F(x) generally overestimates the range $\{F(\tilde{x}) \mid \tilde{x} \in x\}$, there may or may not be a solution point in x if $0 \in F(x)$. In this case we try to speed up the algorithm by attempting to find a smaller box x' containing all solutions in x.

We illustrate the method in the case m=1, n=2, where planar curves described by one equation in two unknowns are sought. In this case, the problem is simpler since it can be treated without using linear algebra. For the sake of clarity we write our equation, restricted to a

2-dimensional box, as

$$f(\tilde{x},\tilde{y}) = 0,\ \tilde{x}\in x,\ \tilde{y}\in y,$$

where $x,y \in \mathbb{IR}$, and denote by f_x and f_y interval extensions of the partial derivatives with respect to the first and second variable, respectively. The mean value theorem gives

$$0 = f(\tilde{x},\tilde{y}) = f(\tilde{x},\underline{y})+f_y(\tilde{x},\eta)(\tilde{y}-\underline{y})$$
$$= f(\underline{x},\underline{y})+f_x(\xi,\underline{y})(\tilde{x}-\underline{x})+f_y(\tilde{x},\eta)(\tilde{y}-\underline{y})$$

for suitable $\xi\in x$, $\eta\in y$, hence

$$\tilde{a}(\tilde{x}-\underline{x})+\tilde{b}(\tilde{y}-\underline{y})+\tilde{c} = 0 \text{ for some } \tilde{a}\in a,\ \tilde{b}\in b,\ \tilde{c}\in c, \tag{2.2}$$

where

$$a = f_x(x,y),\ b = f_y(x,y),\ c = f(\underline{x},\underline{y}). \tag{2.3}$$

On the computer, c will also be a small interval, due to outward rounding interval arithmetic. If $0\notin a,b$ then (2.2) implies

$$\tilde{x} \in x' = \left(\underline{x} - \frac{b(y-\underline{y})+c}{a}\right) \cap x, \tag{2.4a}$$

$$\tilde{y} \in y' = \left(\underline{y} - \frac{a(x-\underline{x})+c}{b}\right) \cap y, \tag{2.4b}$$

and often, x',y' are considerable improvements over x,y. In particular, this is likely the case when x and y are narrow intervals since then the intervals a,b defined in (2.3) will also be narrow. If $0\in a$ or $0\in b$ then we cannot use both formulae of (2.4) but we can exploit (2.2) by using the ternary operator Γ defined in (1.3). We get the enclosures

$$\tilde{x} \in x' = \underline{x}-\Gamma(a,b(y-\underline{y})+c,\underline{x}-x), \tag{2.5a}$$

$$\tilde{y} \in y' = \underline{y}-\Gamma(b,a(x-\underline{x})+c,\underline{y}-y). \tag{2.5b}$$

Since $\underline{x}-x,\underline{y}-y \le 0$, the evaluation of Γ takes at most one real division. If it turns out that one of the intervals x' or y' computed by (2.4) or (2.5) is empty, the box (x,y) contains no solution point and can be discarded; otherwise the algorithm proceeds with (x',y') in place of (x,y).

In the general case we have to consider the problem

$$F(\tilde{x}) = 0,\quad \tilde{x}\in x \tag{2.6}$$

for a subbox x of the initial box x^0. Using a center $\check{z}\in x$ (which takes the place of $(\underline{x},\underline{y})$ in the above consideration of the case m=1, n=2), we have for each $\tilde{x} \in \Sigma(F,x)$ and each i= 1,...,m a relation of the form

(2.7) $0 = f_i(\tilde{x}) = F_i(\tilde{z}) + F'(\xi^i)(\tilde{x}-\tilde{z})$ for some $\xi^i \in x$.

Useful choices for the center $\tilde{z}$ are the midpoint or one the the corners of x. Let us write

$$\tilde{d} := \begin{pmatrix} \tilde{x}-\tilde{z} \\ 1 \end{pmatrix}, \qquad \tilde{A} := \begin{pmatrix} F_1'(\xi), & F_1(\tilde{z}) \\ F_m'(\xi^m), & F_m(\tilde{z}) \end{pmatrix},$$

so that

(2.8) $\tilde{d} \in d := \begin{pmatrix} x-\tilde{z} \\ 1 \end{pmatrix}, \qquad \tilde{A} \in A := (F'(x), F(\tilde{z}))$;

note in particular that $d \ni 0$ if the center is chosen as $\tilde{z} = \underline{x}$. Using (2.8), we can formulate (2.7) as a homogeneous linear interval equation

(2.9) $\tilde{A}\tilde{d} = 0, \quad \tilde{A} \in A, \ \tilde{d} \in d.$

(Note that $d \ni 0$ if $\tilde{z} = \underline{x}$.) If this linear interval system is found incompatible then x can be discarded; the homogeneous formulation (2.9) is preferred to the inhomogeneous formulation

$$A'(\tilde{x}-\tilde{z}) = -F(\tilde{z}), \quad A \in F'(x)$$

in order to enhance the recognition of incompatible systems for ill-conditioned Jacobi matrices F'(x).

Homogeneous linear interval equations are considered in Section 3; apart from tests for inconsistency we shall generalize (2.4), (2.5) to construct an interval vector $d' \subseteq d$ such that (2.9) implies $\tilde{d} \in d'$. In this case, the active part of the box x is contained in the subbox $x' = \tilde{z} + d'$, and we may replace x by x' without losing the covering property of the set of boxes.

The derivative enclosures required in (2.8) can be calculated recursively in PASCAL-SC (Rall [18]), requiring arithmetical expressions for the F_i (but not for their derivatives). In fact it is more efficient to replace the derivatives by interval slopes which satisfy similar recursions (Krawczyk and Neumaier [9], Neumaier [16])) and yield a matrix A also satisfying (2.9), but with narrower coefficients than those obtainable by (2.8).

(iii) Check for empty or small box. After having tried to make the box smaller in (ii), we might have succeeded in discarding the whole box; in this case we proceed with the next box from the stack. Or we might have reduced the box to such small size that the specified accuracy requirements are satisfied; in this case the box is printed (or marked on a screen) and then discarded. As criterion for a box x to be small we used in our implementation

(2.10) $\varrho(x_i) \le \varrho(x_i^0)/res_i$ for $i = 1,\dots,n$,

with integers res_i specifying the resolution; an additional test could be based on function values, printing and discarding a box x when the enclosure $F(x) \cap Ad$ for the range of F over x is small,

(2.11) $|f(x) \cap Ad| \le \eta$,

with a small number η specified in advance.

We emphasize that, while every solution point of (2.1) is contained in some of the boxes printed (if the algorithm stops with empty stack), it is not guaranteed that every box printed contains a solution point; also the argument in Section 1 which showed that every $\tilde{x}$ with $F(\tilde{x}) \ne 0$ was discarded at some stage depends on arbitrarily fine subdivision, i.e. holds only for $\varepsilon, \eta \to 0$. However, it is possible to derive sufficient conditions for the existence of a solution point in a box x, similar in spirit to the existence tests discussed in Qi [17] and Neumaier [16].

(iv) Bisection. If the reduced box is neither empty nor small, the box is split into two boxes which are then replaced into the stack. To guarantee finite termination and to enhance narrow intervals in the entries of (2.8), it is sensible to bisect the box in the coordinate with the largest radius. So x is split into x^1 and x^2, where

$$x_k^1 = [\underline{x}_k, \check{x}_k], \quad x_k^2 = [\check{x}_k, \overline{x}_k],$$

$$x_i^1 = x_i^2 = x_i \text{ if } i \ne k,$$

and the index k satisfies $\varrho(x_k)/res_k \ge \varrho(x_i)/res_i$ for all $i \ne k$. Clearly, after $n \cdot s$ nested bisections, $r(x) := \max\{\varrho(x_1)/res_1, \dots, \varrho(x_n)/res_n\}$ has been reduced to $\le 2^{-s} r(x^0)$; hence after at most $n \cdot \log_2(r(x^0)/\varepsilon)$ nested bisections, a subbox is discarded. Since the active stack size (given

by the variable *last*) equals the number of nested bisections used to create the current box, it follows that a stack size of $O(n \log \varepsilon^{-1})$ is sufficient to store all intermediate boxes. A more detailed count shows that in fact a stack with

$$bmax = 1 + \sum_{i=1}^{n} \log_2 \frac{\varrho(x_i^0)}{\varepsilon res_i}$$

boxes is enough. For small ε, this worst case bound is rarely achieved, and often $bmax = 1+(n+1-m)\cdot\log_2(r(x^0)/\varepsilon)$ is a more reasonable choice.

3. Homogeneous linear interval equations

In this section we consider the homogeneous linear interval equation

(3.1) $\quad \tilde{A}\tilde{x} = 0, \quad \tilde{A}\in\mathbf{A}, \ \tilde{x}\in\mathbf{x},$

where $\mathbf{A}\in\mathbb{IR}^{m\times n}$ and $\mathbf{x}\in\mathbb{IR}^n$, $m\le n$. (This covers the problem (2.9) with d in place of x and n+1 in place of n). We are interested in interval enclosures of the solution set of (3.1), defined as

(3.2) $\quad \Sigma := \{\tilde{x}\in\mathbf{x} \mid \tilde{A}\tilde{x}=0 \text{ for some } \tilde{A}\in\mathbf{A}\},$

and in particular in conditions which guarantee that (3.1) is inconsistent, i.e. $\Sigma=\emptyset$. By a result of Beeck [3], the solution set has the simpler characterization

(3.3) $\quad \Sigma = \{\tilde{x}\in\mathbf{x} \mid 0\in\mathbf{A}\tilde{x}\}.$

In general, Σ has a very complicated shape, and the smallest interval enclosure $x_{opt} = \square\Sigma$ seens to be very difficult to find. Even the decision whether $\Sigma=\emptyset$ or not seems to require exponentially many operations. However, if the entries of x have constant sign then Σ is a convex polyhedron in $\mathbb{R}^n$ (Beeck [3]), and in this case, our problem can be viewed as a multi-objective linear programming problem. We show this for the case $x\ge 0$. Here $\mathbf{A}\tilde{x} = [\underline{\mathbf{A}}\tilde{x}, \overline{\mathbf{A}}\tilde{x}]$ whence

(3.4) $\quad \Sigma = \{\tilde{x}\in\mathbb{R}^n \mid \underline{x}\le\tilde{x}\le\bar{x}, \ \underline{\mathbf{A}}\tilde{x}\le 0\le\overline{\mathbf{A}}\tilde{x}\} \ (\text{if } x\ge 0)$

is defined by linear inequalities. Therefore standard linear programming techniques can be used to decide whether $\Sigma=\emptyset$, and if this is not the case, the 2n problems

$$\text{minimize } \pm\tilde{x}_i \text{ subject to } \tilde{x}\in\Sigma$$

determine the lower and upper bounds of the components of x_{opt}. However, a rigorous treatment in the presence of rounding errors is more complicated (cf. Krawczyk [8], Jansson [5]), and we have not used this approach.

Thus, we content ourselves with sufficient conditions for $\Sigma=\emptyset$, and for less than optimal interval enclosures of Σ. A simple test can be obtained by looking at each component separately. From (3.1) we find

$$0 = \sum_{k=1}^{n} \tilde{A}_{ik}\tilde{x}_k \in \sum_{k=1}^{n} \mathbf{A}_{ik}\mathbf{x}_k$$

and

$$\tilde{A}_{ij}\tilde{x}_j = -\sum_{k\ne j} \tilde{A}_{ik}\tilde{x}_k \in -\sum_{k\ne j} \mathbf{A}_{ik}\mathbf{x}_k.$$

Using again the ternary operator Γ defined in (1.3), we get the improved enclosure

(3.6) $\quad \tilde{x}_j \in \Gamma\left(a_{ij}, -\sum_{k\ne j} \mathbf{A}_{ik}\mathbf{x}_k, x_j\right) \quad (j=1,\dots,n).$

To reduce the work needed to compute (3.6) we write

$$p_j := \mathbf{A}_{ij}\mathbf{x}_j, \quad s_j := \sum_{k\ne j} \mathbf{A}_{ik}\mathbf{x}_k, \quad s := \sum_{k=1}^{n} \mathbf{A}_{ik}\mathbf{x}_k = s_j + p_j.$$

For all j, we then have $\underline{s}=\underline{s}_j+\underline{p}_j$, $\bar{s}=\bar{s}_j+\bar{p}_j$, hence

$$-s_j = [-\bar{s}_j, -\underline{s}_j] = [\bar{p}_j-\bar{s}, \underline{p}_j-\underline{s}].$$

Therefore the improved enclosure (3.6) can be computed for fixed i in O(n) operations by the program piece

```
good := true; s := 0
for j := 1 to n do
begin p[j] := A[i,j]*x[j]; s := s+p[j] end;
good := 0 in s;
if good then for j := 1 to n do
begin minuss.inf := p[j].sup -< s.sup;
      minuss.sup := p[j].inf -> s.inf;
      gamma (A[i,j],minuss,x[j],good) end;
```

where *gamma* (a,b,x,*good*) denotes a procedure which sets *good* := *false* if $\Gamma(a,b,x) = \emptyset$, and otherwise overwrites x by $\Gamma(a,b,x)$. Clearly, we can use each equation in turn for a repeated improvement of the box.

Because of the analogy with the classical Gauss-Seidel iteration method for linear equations we call this improvement method the generalized Gauss-Seidel method, and refer to going with (3.6) through each equation once as one sweep of the method. Applied directly to (3.1), the generalized Gauss-Seidel method is usually not very efficient, but the performance can improve drastically when the linear system (3.1) is preconditioned by multiplying it on the left with a real m×m-matrix C. This yields the system

(3.7) $\tilde{A}'\tilde{x} = 0, \quad \tilde{A}'=C\tilde{A}\in CA, \ \tilde{x}\in x,$

and the generalized Gauss-Seidel method is now applied to (3.7), i.e. with CA in place of A. As yet we have no conclusive results about the best choice of C, but in analogy with the case of square inhomogeneous linear interval equations (surveyed in Neumaier [15]), it seems plausible to take C as an approximate inverse of a matrix $\tilde{B}$ composed of m linearly independent columns of some matrix $\tilde{A}\in A$. In our program we chose $C=\tilde{B}^{-1}$, using row pivoting to select $\tilde{B}$, and then performed two sweeps of the generalized Gauss-Seidel method to improve the box.

We end this section by explaining why the homogeneous approach has an advantage over the inhomogeneous formulation. Consider the square inhomogeneous system

(3.8) $\tilde{B}\tilde{z} = \tilde{b}, \quad \tilde{B}\in B, \ \tilde{b}\in b, \ \tilde{z}\in z,$

where $B\in\mathbb{IR}^{m\times m}$, $b,z\in\mathbb{IR}^m$ and the related homogeneous system

(3.9) $\tilde{A}\tilde{x} = 0, \quad \tilde{A}\in A := (B,-b), \ \tilde{x}\in x:=\begin{pmatrix} z \\ 1 \end{pmatrix}.$

To make the point, consider the special case of (3.8) where B and b are thin, B is singular of rank m-1, and the system is inconsistent, i.e. (B,-b) has rank m. (In general, the situation is similar whenever (3.8) is inconsistent and B is singular or ill-conditioned.) Then the standard way of solving (3.8) by preconditioning with an approximate midpoint inverse C leads (due to roundoff) to a matrix C with huge elements, and all information in (3.8) is lost after multiplying by C. On the other hand, row pivoting assures that in (3.9) one of the columns of B is replaced by -b, thus leading to a reasonable C, and generalized Gauss-Seidel will (at least in the thin case) almost certainly detect the inconsistency in the second sweep.

4. Numerical examples.

In this section we report on some calculations done with the MS-DOS version [1] of PASCAL-SC on an Olivetti M24. In most cases we list the number f of boxes considered (for each box, one function+slope call and one linear system solve is required), the number b of boxes in the final covering, the maximal stack size s, and the execution time t.

Example 1. Pairs of cubic equations of the form

$$\alpha_1x_1^2x_2+\alpha_2x_1^2+\alpha_3x_1x_2+\alpha_4x_1+\alpha_5x_2 = 0,$$
$$\alpha_6x_1^2x_2+\alpha_7x_1x_2^2+\alpha_8x_1x_2+\alpha_9x_2^3+\alpha_{10}x_2^2+\alpha_{11}x_2+\alpha_{12} = 0$$

arise in combustion chemistry. In the box $x_1\in[0,1]$, $x_2\in[0,1]$, we find for the coefficients given in Morgan and Shapiro [11] a single solution box

$$x_1=6.849808^{785}_{590}*10^{-6}, \ x_2=1.0237169^{333}_{075}*10^{-3}.$$

(The results do not agree with those of [11], probably due to some misprint there.) We have f=342, b=1, s=29, t=11m15s when $res_1=res_2$, but with $res_1=1000res_2$ — reflecting improved scaling of the problem —, the answer is found much quicker (f=59, b=1, s=24, t=2m21s). This illustrates the need to develop an automatic, self-scaling bisection procedure.

Example 2. Robot manipulation leads to systems of 8 polynomial equations in 8 unknowns. With the data from Morgan and Shapiro [11], we find 16 solutions in the initial box $x_i\in[-1,1]$ (i=1,...,8), covered by 16 boxes. We have f=240, b=16, s=8, t=53m05s. Sample coordinates for the first box found:

$$x_1= 0.1644316658^{7}_{5}, \ x3=-0.9547284344^{91}_{82},$$
$$x_5=-0.9111547^{644}_{582}, \ x_7= 0.991322415^{13}_{08}.$$

Example 3. The 1-dimensional manifold defined by

$$x^3-xy^2+x^2-xy-y^2 = 0, \quad x\in[-5,5], \ y\in[-3,3]$$

is, for different resolutions, covered by the set of boxes shown in the introduction (Fig.1). The covering method correctly detects three curves, two of which intersect. For the highest resolution, f=1043, b=808, t=15m33s.

Example 4. Ushida and Chua [24] consider the following pair of equations for a simple tunnel diode circuit:

(4.1) $0.43v_1^3-2.89v_1^2+4.56v_1 = 2.5v_2^3-10.5v_2^2+11.8v_2 = i,$

(4.2) $v_{in} = v_1+v_2+3.2i.$

The covering method for (4.1) correctly detects two components curves one of them closed, and the subsequent calculation of v_{in} from (4.2) leads to the transfer characteristics shown in Fig.2. For (4.1), we have f=1021, b=883, t=14m33s, and the transformation (4.2) takes 10 further minutes.

Example 5. The polynomial system

$$z = x^2+y^2, \qquad az = y^2+z^2$$

has as solution set a figure 8 curve if a<1, two touching closed curves if a=1, and an isolated point (at zero) plus a closed curve if a>1. The projection to the x-y plane is a so-called hippopede (Lawrence [10]). For a=1.1 and $x\in[-1.5,1.5]$, $y\in[-1,1]$, $z\in[0,4]$ we get a covering whose projection to the x-y plane is shown in Fig.3. (The spurious boxes are due to the influence of the singularity at zero, and disappear when the resolution is increased). We have f=1719, b=641, s=14, t=55m10s.

Example 6. Rheinboldt [19] describes a system of 5 polynomial equations in 8 unknowns $y_1,\ldots,y_5,u_1,\ldots,u_3$ arising in aircraft equilibrium problems. For $u_1=0.1$, $u_3=0$ we get a covering of drawing accuracy with f=1518, b=588, s=14, t=4h33m20s

We note that for curve coverings (n=m+1, Examples 3-6) the number of boxes considered was less than 3 times the number of boxes in the final covering. The time required for each box considered is roughly proportional to n^2, so that the $O(n^3)$-complexity for solving linear systems is not yet observed for small n.

Example	m	n	f/b	$t/(f\cdot n^2)$
1	2	2	342.00	0.493s
2	8	8	15.00	0.207s
3	1	2	1.29	0.224s
4	1	2	1.16	0.214s
5	2	3	2.68	0.214s
6	5	6	2.58	0.300s

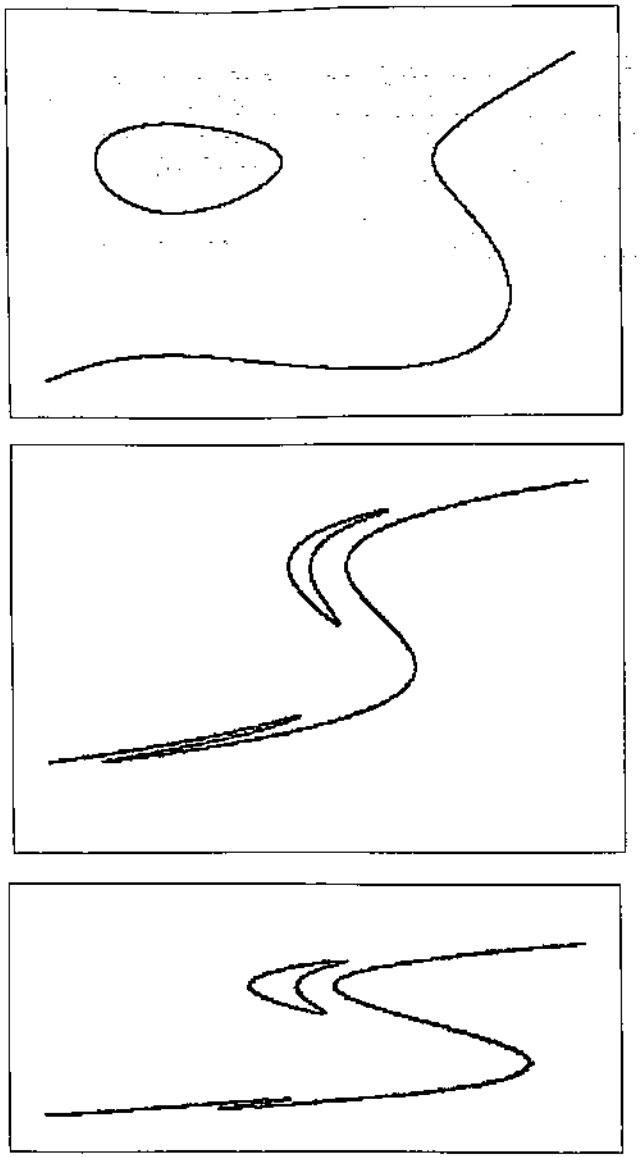


Fig. 2: (a) v_1 versus v_2; (b) v_{in} versus v_1; (c) v_{in} versus v_2.

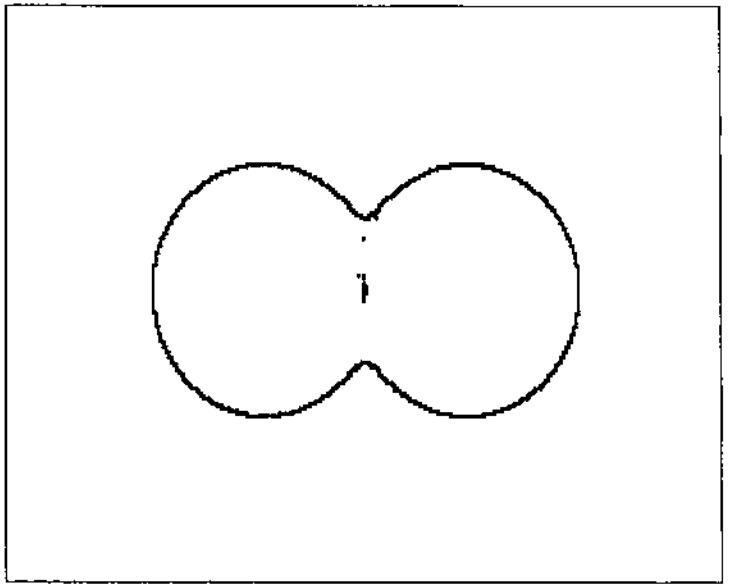


Fig. 3: A covering for the hippopede.

Some of the examples were also tried on other microcomputers; the following speed-up times were observed on Examples 2 and 5:

6.5	ATARI
6 - 7	KWS SAM68K (without BAP)
12 - 15	KWS SAM68K (with BAP)
12	KWS EB68/20 (without BAP)

5. Final remarks

All systems solved above were low-dimensional polynomial systems, and it seems that such systems are solved easily and reliably by the covering method. There is no problem in solving piecewise polynomial systems (with rational expressions involving also abs, max,min) when the slopes are calculated as in Neumaier [16]. This makes the method particularly suitable for problems in computer aided geometrical design (CAGD).

The situation may be different for high dimensional systems – in particular if these involve functions not defined for all values of the variables –, since, due to overestimation, the method may generate initially an exponential number of boxes. (Locally, however, it can be shown that the number of boxes is exponential only in the manifold dimension p and not in m or n.) A simple example of this situation is the system $F(x)=0$, where $F_i(x)=2x_i/(1-x_i+x_i^2)-1$, $x_i\in[0,1]$ $(i=1,\ldots,n)$. In this case a natural approach may be to combine the covering method with continuation techniques to save searching large empty regions: simply choose the next box near a box containing a solution point, taking in account the direction in which the curve leaves the current box.

References

1. U. Allendörfer et al., PASCAL-SC Information Manual and Floppy disks, Wiley - Teubner 1987.
2. Allgower, E.L. and K. Georg, Relationships between deflation and global methods in the problem of approximating additional zeros of a system of nonlinear equations, pp. 31-42 in: Homotopy Methods and Global Convergence, Proc. Nat. Adv. Res. Inst. Porto Cervo, NATO Conf. Ser. II, Syst. Sci. 13 (1983).
3. Beeck, H., Über Struktur und Abschätzungen der Lösungsmengen von linearen Gleichungssystemen mit Intervallkoeffizienten, Computing 10 (1972), 231-244.
4. Bohlender, G. et al., PASCAL-SC: A PASCAL for contemporary scientific computation, IBM Res. Rep. RC9009, Yorktown Heights, New York 1981.
5. Jansson, Ch., Zur linearen Optimierung mit unscharfen Daten, Dissertation, Kaiserslautern 1985.
6. Klatte, R. und J. Wolff von Gudenberg, Forschungsschwerpunkt "Computerarithmetik und Programmiersysteme für ingenieurwissenschaftliche Anwendungen", Inst. f. Angew. Math., Univ. Karlsruhe 1986.
7. Koparkar, P.A. and S.P. Mudur, Subdivision techniques for processing geometric objects, pp. 751-801 in: Fundamental algorithms for computer graphics (ed. R.A. Earnshaw), Springer, Berlin 1985.
8. Krawczyk, R., Fehlerabschätzung bei linearer Optimierung, pp. 215-222 in: Interval Mathematics (ed. K. Nickel), Springer Lecture Notes Comp. Science 29 (1975), 215-222.
9. Krawczyk, R. and A. Neumaier, Interval slopes for rational functions and associated centered forms, SIAM J. Numer. Anal. 22 (1985), 604-616.

10. Lawrence, J.D., A catalog of special plane curves, Dover, New York 1972.

11. Morgan, A. and V. Shapiro, Box-bisection for solving second-degree systems and the problem of clustering, ACM Trans. Math. Softw. 13 (1987), 152-167.

12. Moore, R.E., Methods and Applications of Interval Analysis, SIAM, Philadelphia 1979.

13. Mudur, S.P., P.A. Koparkar, Interval methods for processing geometric objects, Comput. Graphics Appl. 3 (1984), 7-17.

14. Neumaier, A.,, New techniques for the analysis of linear interval equations, Linear Algebra Appl. 58 (1984), 273-325.

15. Neumaier, A., Linear interval equations, pp. 109-120 in: Interval Mathematics 1985 (ed. K. Nickel), Springer Lecture Notes Comp. Sci. 212 (1986).

16. Neumaier, A., Existence of solutions of piecewise differentiable systems of equations, Arch. Math. 47 (1986), 443-447.

17 Qi, L., A note on the Moore test for nonlinear system, SIAM, J. Numer. Anal. 19, 845-850 (1982).

18. Rall, L.B., Differentiation in PASCAL-SC: type gradient, ACM. Trans. Math. Softw. 10, 161-184 (1984).

19. Rheinboldt, W.C., Computation of critical boundaries on equilibrium manifolds; SIAM J. Numer. Anal. 19 (1982), 653-669.

20. Rheinboldt, W.C., Numerical analysis of parametrized nonlinear equations, Wiley, New York 1986.

21. Rheinboldt, W.C., Error questions in the computation of solution manifolds of parametrized equations, These proceedings.

22. Schwetlick, H., Numerische Lösung nichtlinearer Gleichungen, Oldenbourg, München-Wien 1979.

23. Toth, D.L., On ray tracing parametric surfaces, Computer graphics 19 (1985), 171-179.

24. Ushida, A. and L.O. Chua, Tracing solution curves of nonlinear equations with sharp turning points, Circuit Theory Appl. 12 (1984), 1-21.

25. Zangwill, W.I. and C.B. Garcia, Pathways to solutions, fixed points, and equilibria, Prentice-Hall, Englewood Cliffs, N.J., 1981.