Hybrid Interval Marching / Branch and Bound Method for Parametrized Nonlinear Systems

R. BAKER KEARFOTT (rbk@louisiana.edu) and MIHYE KIM (mxk0013@louisiana.edu)
U.L.Lafayette Box 4-1010, Lafayette, Louisiana, 70504-1010, U.S.A.

October 20, 2004

Abstract. The hybrid interval marching / branch and bound method for parametrized nonlinear systems presented in this paper can find *all* the solution components (curves) with mathematical certainty within a given search region of n-space. Kearfott and Xing introduced an interval step control method which improved on floating point step controls with a mathematical guarantee that the predictor algorithm will not jump from one path to another. But those methods did not try to detect all the solution components (curves) in a search region, and also their algorithms depend on heuristics to adjust box sizes. The algorithms for our hybrid interval marching / branch and bound method for parametrized nonlinear systems not only can find all global solution components of n-dimensional parametric curves but also depend on the geometry of a function rather than heuristics to set the box size. Snyder's implicit curve algorithm, which employs subdivision and global parameterizabability, can find the global solution curves of nonlinear parametric equations, but his method was just 2-dimensional. Our algorithm is designed to handle n-dimensional systems of nonlinear parametric equations. We successfully ran our hybrid interval marching / branch and bound method for several test functions.

Keywords: interval marching method, branch and bound method, solution components, interval step control method, predictor algorithm, global solution curves

AMS subject classification: 65G40, 65H10, 65H20

1. Introduction

We have implemented and run for the first time a hybrid interval marching / branch and bound method to find *all* solution components (curves) S_x to parametrized nonlinear systems in an n-dimensional search region where S_x is,

$$S_x = \{x = (z, \lambda) \in \mathbb{R}^n \times \mathbb{R} \mid \; H(x) = 0\}, \text{where}\, H : \mathbb{R}^{n+1} \longrightarrow \mathbb{R}^n.$$

We introduce a branch and bound method for parametrized nonlinear systems. We start the process by forming composite boxes and complement boxes[1]. These composite and complement boxes are designed to guarantee that all components of S_x in the n-dimensional search region

[1] Refer to Kearfott's box complementation algorithms in [4, page 154].

© 2004 *Kluwer Academic Publishers. Printed in the Netherlands.*

have been found. First, we construct the composite boxes, which are unions of solution boxes, and then construct the complement boxes around them, and search each complement box for the presence of a new point at which $H(x) = 0$, thus a new solution component. If a new solution component is found, then we execute the interval step control method again; otherwise, we bisect the box until we find a new component of S_x.

Our work in this paper is different from previous work in the following ways. The interval step control method of Kearfott and Xing [1] found only one component of the solution set S_x to the parametrized nonlinear systems, whereas we *rigorously* find *all* components within a given *n-dimensional* region.

Snyder [5] used only subdivision, and not marching, in his work on implicit curve algorithms, which employs subdivision and global parameterizabability of an interval (or a rectangle) through which a solution curve passes. Also, his method was basically just 2-dimensional while our method is n-dimensional.

Throughout the paper, boldface denotes intervals, lower case denotes vectors, and upper case denotes matrices and functions.

1.1. The Mathematical Problem

The mathematical problem we are solving in this paper is: Given a nonlinear parametric equation $H : \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}^n$, we find, with mathematical certainty, $H(x) = 0$ within a search region of n-space, where $x = (z, \lambda) \in \mathbb{R}^n \times \mathbb{R}$ is a regular point[2] of H. Since the Jacobian matrix $H'(x)$ is non-singular, $\exists$ a solution curve $c(s)$ and a tangent vector or a null vector $\dot{c}(s)$ with a specific direction, where arc length $s \in I$, and I is an open interval around x.

Once we solve $H(x)$ for $x = (z, \lambda)$ by the interval predictor-corrector method, we form $(n+1)$ coordinate intervals centered at x, i.e., we form an $(n + 1)$-dimensional box centered at x. This is to apply interval arithmetic to verify the existence and the uniqueness of a solution curve inside the box whose coordinate intervals are[3] $\boldsymbol{x}_i$, by a validation technique such as the interval Gauss-Seidel method[4]. A validation tool like the interval Gauss-Seidel method plays a vital role in verifying the existence and uniqueness of the solution component S_x in interval computation. We perform the interval Gauss-Seidel method for k iterations,

[2] x is a regular point of H if the Jacobian matrix $H'(z, \lambda)$ is of full rank n, where $z \in \mathbb{R}^n$ and $\lambda \in \mathbb{R}$.

[3] $\boldsymbol{x}_i$ is the interval variable of each i coordinate of a $(n + 1)$-dimensional box for $1 \le i \le n + 1$.

[4] Refer to the interval Gauss-Seidel method in [1].

for some positive integer k, until the error tolerance[5] $|\ \tilde{\boldsymbol{x}}_i - \boldsymbol{x}_i\ | \leq\ 10^{-8}$. Thus if the inclusion[6] $\tilde{\boldsymbol{x}}_i\ \subset \mathrm{int}(\boldsymbol{x}_i)$ is[7] successful for each i, then the existence and the uniqueness of each solution coordinate x_i in the box $\boldsymbol{x}_i$ is verified (Theorem 2).

Detailed explanations of our techniques are given in individual algorithms, which are available from the author.

2. Summary of Our Hybrid Interval Marching / Branch and Bound Method

The motivation for interval step control instead of previous floating point algorithms is related to avoiding problems related to curve following. Earlier approximate step control methods work well for smooth and isolated curves, but will jump from one path to another if there are many paths near some points, or can erroneously reverse orientation if rapid changes in curvature occur along the path [1]. Another limitation of earlier marching methods is that they can skip some branches of the intersection curve. Our present hybrid interval marching / branch and bound method for parametrized nonlinear systems both guarantees that the predictor algorithm will not jump from one path to another and guarantees that all solution paths are then found. This is due to the *uniqueness* guaranteed by the interval Gauss-Seidel method. Our algorithms detect all solution components within the original region $\boldsymbol{x}$ whenever there is more than one such curve.

Our algorithm is composed of five main parts:

1. Finding solution components (curves) S_x
2. Validating the solution components S_x by the interval Gauss-Seidel method
3. Forming the composite boxes and the complement boxes
4. Finding points at which $H(x) = 0$ for the new components
5. Evaluating the complement boxes

The old method, which I modified, depended on heuristics of step control [1]. Our improved method depends on the natural geometry of the functions rather than heuristics. Instead of adjusting the stepsize s, we use the local values of a given function H and a tangent line at

[5] $\tilde{\boldsymbol{x}}_i$ is the smallest possible interval which contains a solution point.
[6] Refer to the definition of inclusion in Definition 1
[7] $\mathrm{int}(\boldsymbol{x}_i)$ denotes the interior of the interval $\boldsymbol{x}_i$.

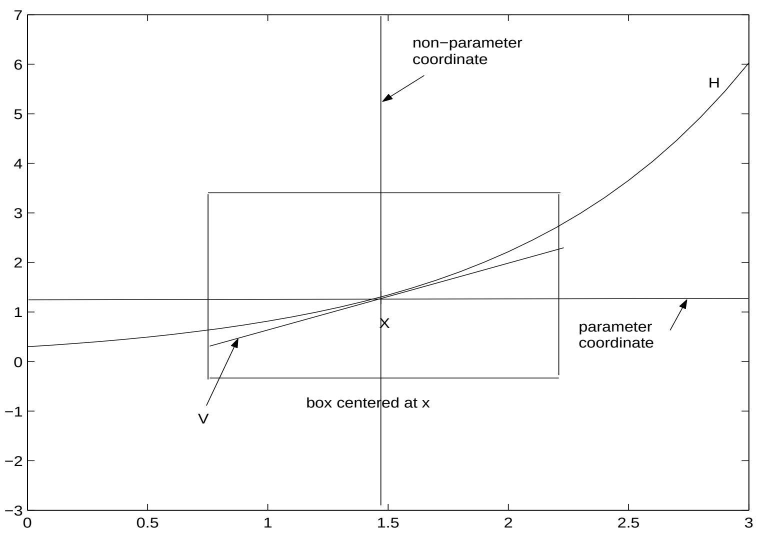

Figure 1. Choosing the parameter p as the largest coordinate of a null vector v

the corrector point $\tilde{x}^+$ to calculate the proper shape vector η, where shape vectors are non-parameter coordinate vectors, $\boldsymbol{x}_{\neg p}$.

2.1. Setting the Box Coordinate Bounds

In constructing a box around a solution $\check{x}$, we calculate η from the multi-variate one degree Taylor polynomial H. Thus we can form the box following the local geometry of the function H.

We calculate the null vector v, which is orthogonal to the rows of the Jacobi matrix of H at a solution point, $\check{x}$. The tangent line at $\check{x}$ is in the direction of v, and we choose the *largest* coordinate of the null vector (tangent vector) as the parameter coordinate. This will make the iteration under the interval Gauss-Seidel method faster, because the image $\tilde{\boldsymbol{x}}_{\neg p}$ of the non-parameter coordinates of $\boldsymbol{x}_{\neg p}$ will be smaller in that case. This is clear from figure 1, since the solution curve is almost parallel to the parameter coordinate, and it will make the convergence of the sequence of the non-parameter coordinates $\boldsymbol{x}_{\neg p}(i)$ of the box $\boldsymbol{x}$ such that the inclusion $\tilde{\boldsymbol{x}}_i \subset \text{int}(\boldsymbol{x}_i)$ will happen easier. Note that figure 1 shows that the box is constructed so that the image $\tilde{\boldsymbol{x}}_i$ must contain the entire curve.

In our method, we are using the geometry of a given function to construct a box. We calculate $\boldsymbol{x}_{\neg p}$ from a multivariate degree one Taylor polynomial. The multivariate degree one Taylor polynomial for the

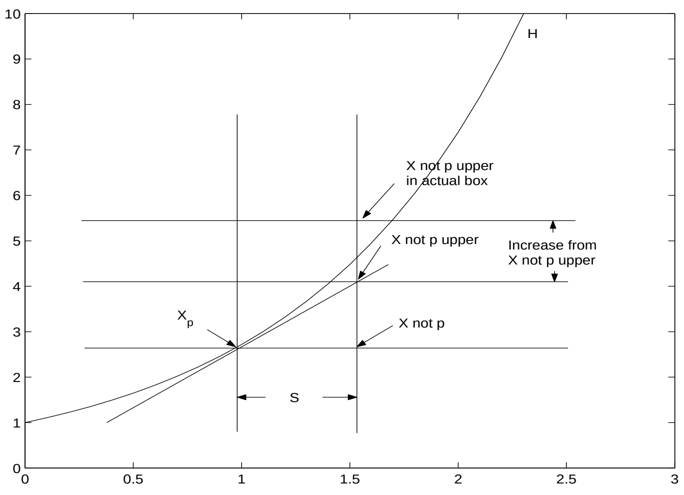

Figure 2. Setting non-parameter coordinates of a box

function H is written as:

$$H(x) \approx H(\check{x}) + H'(\check{x})(x - \check{x}) = 0. \qquad (1)$$

If we fix $\underline{x_p} = \overline{x_p}$, we get an $n \times n$ system of linear equations. Then, from equation 1,

$$\begin{aligned} H(\check{x}) &+ H'(\check{x})(x - \check{x}) \\ &= H(\check{x}) + \sum_{i=1}^{n+1} H'(:,i)(x - \check{x}) \\ &= H(\check{x}) + H'(:,p)(\overline{x_p} - \check{x}_p) + \sum_{\substack{i=1 \\ i \neq p}}^{n+1} H'(:,i)(x - \check{x}) \\ &= H(\check{x}) + H'(:,p)(\overline{x_p} - \check{x}_p) + H'_{\neg p}(x - \check{x})_{\neg p} \qquad (2) \\ &= H(\check{x}) + H'(:,p)(\overline{x_p} - \check{x}_p) + H'_{\neg p}\overline{x}_{\neg p} - H'_{\neg p}(\check{x})_{\neg p} = 0 \end{aligned}$$

Interchange the terms of equation 3 to get

$$H'_{\neg p}\overline{x}_{\neg p} = H'_{\neg p}(\check{x})_{\neg p} - H(\check{x}) - H'(:,p)(\overline{x_p} - \check{x}_p) \qquad (3)$$

Now we solve Equation 3 for $\overline{x}_{\neg p}$.

Figure 2 shows $\tilde{x}_{\neg p}$ in a box formed by the setting box algorithms after the above calculation.

Validation of existence and uniqueness of the solution set is accomplished by the interval Gauss-Seidel method. We start from $\boldsymbol{x}$

and we replace $\boldsymbol{x}$ by $\tilde{\boldsymbol{x}}$ only if each inclusion $\tilde{\boldsymbol{x}}_i \subset \mathrm{int}(\boldsymbol{x}_i)$ occurs. If $\mid \tilde{\boldsymbol{x}}_i - \boldsymbol{x}_i \mid \leq 10^{-8}$ for $1 \leq i \leq n+1$, then we conclude by the next theorem that the sequence of $\boldsymbol{x}_i$ converges to $\tilde{\boldsymbol{x}}_i$ and that the verification is successful.

THEOREM 1. *Theorem5.1.7 citeneumaierimse90 Assume $F : \mathbb{R}^n \to \mathbb{R}^n$ be Lipschitz continuous on $D \subseteq D_0$, and let A be a regular*[8] *Lipschitz set on D. If $\check{x} \in \boldsymbol{x} \in D$, then every $\boldsymbol{x}' \in \mathbb{IR}$ satisfying*
$N(\boldsymbol{x}, \check{x}) := \check{x} - A^H F(\check{x}) \subseteq x'$
has the following properties:

i. Every zero $x^* \in \boldsymbol{x}$ of F satisfies $x^* \in \boldsymbol{x}'$.

ii. If $\boldsymbol{x}' \cap \boldsymbol{x} = 0$ then F contains no zero in $\boldsymbol{x}$.

iii. If $\check{x} \in \mathrm{int}(\boldsymbol{x})$ and $x' \subseteq \boldsymbol{x}$ then F contains a unique zero in $\boldsymbol{x}$ (and hence in $\boldsymbol{x}'$).

Thus the existence and the uniqueness of x_i in $\boldsymbol{x}_i$ is verified. We prove that the conditions for the above theorem are eventually met for each curve-enclosing box in our algorithm in Theorem 2 below.

In the implementation of the verification process, we have a new advanced heuristic. If the different coordinate widths vary widely, (e.g. $w(x_{i-1}) = 10 * w(x_{i+2})$), it tends to cause more overestimation in interval computation. We solved this problem by replacing the width of each coordinate by max $w(x_i)$ (Figure 2). In our experiments, this method significantly reduced problems rising from overestimation inherent in interval computing. In particular, the method seems to improve the chances that existence and uniqueness will be verified with a larger box. This method is outlined below.

1. Let $\boldsymbol{x}_i$ be each coordinate of a box $\boldsymbol{c}_i$ centered at the corrector point x of a parametric curve $H(x)$ for $1 \leq i \leq n+1$.

2. (Execute the box setting algorithm and the interval Gauss-Seidel method) Each coordinate width $w(x_i)$ is verified for [9] $0 \in \boldsymbol{H}^u(\boldsymbol{x})$.

3. If the inclusion $\tilde{\boldsymbol{x}}_i \subset \mathrm{int}(\boldsymbol{x}_i)$ is not verified, then replace each coordinate width $w(x_i)$ with the maximum of $w(x_i)$ for all i. This will make the verification of the existence and the uniqueness of the solution x in $\boldsymbol{x}$ i.e., $\tilde{\boldsymbol{x}}_i \subset \mathrm{int}(\boldsymbol{x}_i)$ successful by reducing overestimation.

[8] $\boldsymbol{A}$ is regular if every matrix $\tilde{\boldsymbol{A}} \in \boldsymbol{A}$ has rank n.
[9] $f^u(\boldsymbol{x})$ is the range of f over $\boldsymbol{x}$ if f is a function defined over an interval $\boldsymbol{x}$.

For all parametric functions we tried (Chapter 5), the verification with our method was successful. We intend to do further experiments, using this heuristic, with different functions.

2.2. Setting the Predictor Step Size

We have only a few heuristics in our algorithms:

1. We doubled the previous stepsize s when the parameter p changed (and thus the curve changes direction), or when the previous two stepsizes increased during iteration.

2. We halved the stepsize when a box $\boldsymbol{c_b}$ is not verified by the interval Gauss-Seidel method.

Also we set the initial input stepsize to be 0.1, and at the beginning of each subsequent step, we set the stepsize to be the average of the stepsizes so far. Our program is designed so that the function itself will choose the direction of the curve and stepsize.

2.3. On the Box Complementation Process

The application of the box complementation process to the hybrid interval marching / branch and bound method for parametrized nonlinear systems is new. This process completes the process of finding parametric curves in a given region by the interval step control method of Kearfott and Xing [1]. In Kearfott and Xing's paper in 1993 [1], they found a parametric curve by the interval step control method. However, they did not verify the *existence* or the *nonexistence* of the other curves in the region outside the verified region (boxes) along the curve they already found.

Our complementation process, tried for the first time in the context of a parametrized system, made the following possible.

1. Finding a point at which $H(x) = 0$ (approximately) in the complement boxes, which in turn can initiate following a new curve.

2. Removing *efficiently* the part of the given search region through which no curves pass.

3. Finding *all* solutions in the search region, and verifying that all have been found.

We now present descriptions of the various boxes occurring in the search algorithm and how they are formed.

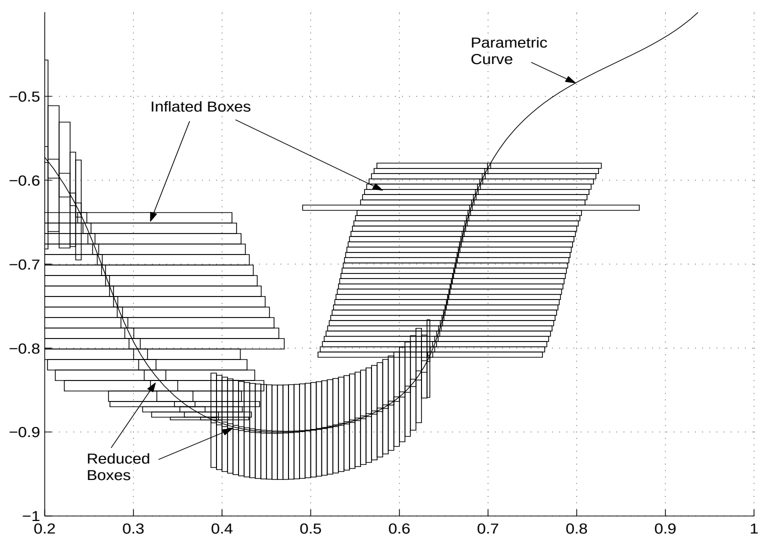

Figure 3. Reduced boxes and inflated boxes of the parametric curve for equation 16

1. reduced boxes $\boldsymbol{r_b}$: Take the verified box, and continue reducing its size by iterating the interval Gauss-Seidel method until no further reduction occurs. As soon as no reduction of size of a box occurs, take the smallest one computed. This will be a reduced box $\boldsymbol{r_b}$. It is ensured that the curve must pass through each reduced box $\boldsymbol{r_b}$ (Figure 3).

2. inflated boxes $\boldsymbol{i_b}$: Perform epsilon-inflation [4] to a constructed box $\boldsymbol{c_b}$. No other curves should pass through each inflated box $\boldsymbol{i_b}$ (Figure 3).

3. composite reduced boxes $\boldsymbol{c_r}$: Form $\boldsymbol{c_r}$ as a union of [10] $\boldsymbol{r_b}$ (Figure 4). Refer to Algorithm 2.

4. composite inflated boxes $\boldsymbol{c_i}$: Form $\boldsymbol{c_i}$ as a union of [11] $\boldsymbol{i_b}$ and take the intersections along the non-parameter coordinates (Figure 4.). Refer to Algorithm 2.

5. complement boxes $\boldsymbol{c_m}$: Form the list of complement boxes $\boldsymbol{l_c}$ (Figure 5).

To see the sequence of box formation, look at Figure 6.

[10] The conditions under which a new union of boxes is started are listed below.

[11] The conditions under which a new intersection of boxes is started are listed below.

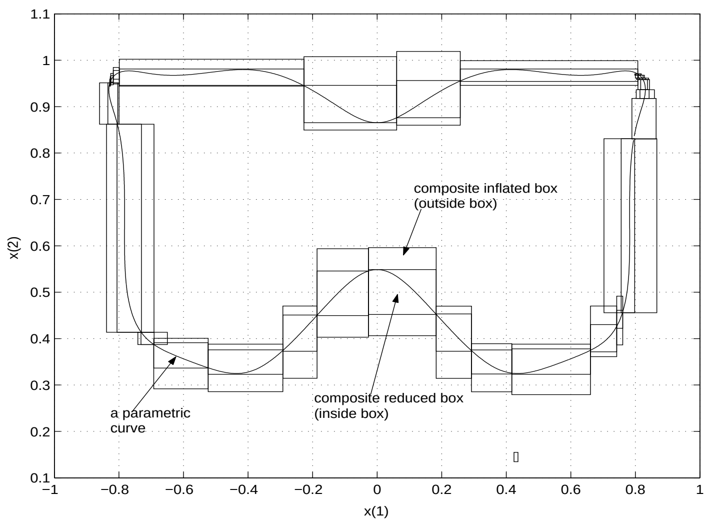

Figure 4. Composite reduced boxes and composite inflated boxes for equation 16

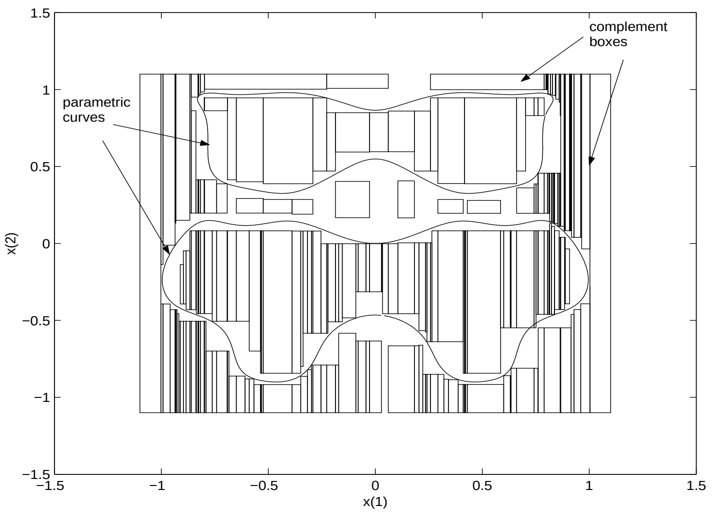

Figure 5. Two curves and complement boxes for equation 16

Forming a new composite box is done under the following conditions:

1. Whenever the parameter changes.

2. When the curve hits the edge of the range box $\boldsymbol{b_0}$ once.

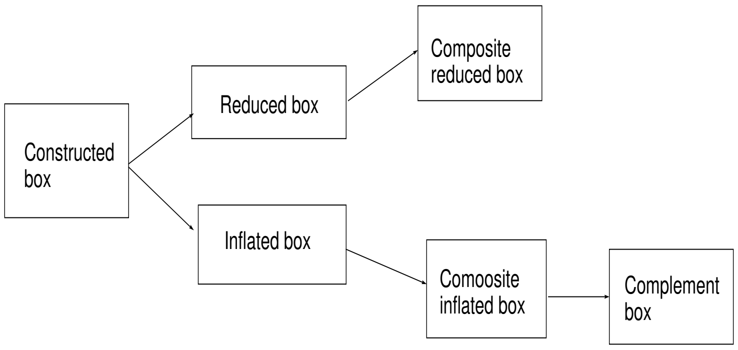

Figure 6. Sequence of Different Box Formation

3. When the difference of the width of $\boldsymbol{c_r}$ is greater than half of the width of $\boldsymbol{c_i}$.

These boxes are formed to make the complementation process easier, since we end up forming fewer complement boxes for the fewer original boxes.

Once the list of complement boxes $\boldsymbol{l_c}$ is formed, each complement box is verified for the *existence* or the *nonexistence* of other curves. Our algorithm can *rigorously* find *all* the curves in the given box $\boldsymbol{b_0}$.

Once more than one curve is found, we need to find a new point at which $H(x) = 0$ to start a new iteration. We check each box in the list of complement boxes, and test the box for solutions to $H(x) = 0$. This is done by checking the intersection of each complement box and *all* the composite inflated boxes $\boldsymbol{c_i}$ for $1 \leq i \leq n$ for the existence of a solution at which $H(x) = 0$.

If we cannot find a solution to $H(x) = 0$, then we bisect the boxes until a certain stopping criterion is satisfied. If there are no more solutions to $H(x) = 0$ at all, we have proved that there are no more curves to follow.

3. Summary of Our Algorithms

The calling diagram for our Fortran implementation appears in Figure 7.

The hybrid interval marching / branch and bound method is a global search, so our algorithm has an integer variable *more_count*, which counts the number of components (curves) found. Our algorithms first find a point w at which $H(w) \approx 0$. We then find a null vector v of

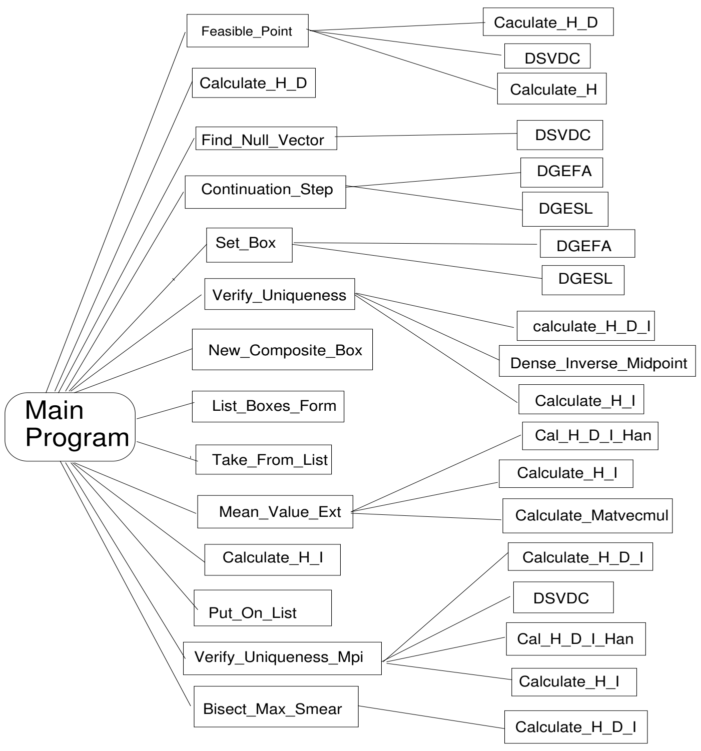

Figure 7. Diagram of Programs

the n by $n+1$ Jacobi matrix $H'(w)$. We take the largest coordinate of v as the parameter coordinate p. The direction in which the curve is followed is made consistent by making the sign of dot product of the present null vector and the previous null vector positive.

We find the non-parameter coordinates $\boldsymbol{x}_{\neg p}$ of approximate solution points along the curve by a predictor-corrector method. Then each solution box $\boldsymbol{x}$ is formed around each solution point $x_{\neg p}$. Each such solution box $\boldsymbol{x}$ is then verified with the interval Gauss-Seidel method to contain a unique solution point $x(x_p)$ for each value of the parameter coordinate x_p within its bounds $\boldsymbol{x}_p$.

To carry out the branch and bound method, we form composite inflated and composite reduced boxes as we explained on page 7, and as is illustrated in Figure 6. After forming the complement boxes (by taking the complement of each composite inflated box with the original range box, using the algorithm in [4, page 154]), we test each complement box $\boldsymbol{x}$ for the presence of solutions of $H(x) = 0$. We use interval extensions of the components of H over $\boldsymbol{x}$ to reject boxes $\boldsymbol{x}$. We now present the algorithms for forming the composite boxes and complement boxes.

Figure 6 represents the sequence of forming different kinds of boxes in our algorithms.

The algorithm for setting boxes, and the algorithm for forming composite boxes are presented below.

ALGORITHM 1. **Set a box around a corrector point**

```
1. Input the corrector point $x^+$, $p$, $H'(x^+)$, and $H(x^+)$.

2. Solve equation 3 for $\overline{x}_{\neg p}$.

3. (Once we get $\overline{x}_{\neg p}$, we set up a box. Let $\kappa$ be any positive integer
   greater than 1 and let $s$ be a steplength, i.e., the width of the parameter coordinate.)
   If $i = p$, then set
      $\boldsymbol{x}_i = [\check{x}_p - s/2,\ \check{x}_p + s/2]$
   else
      $\boldsymbol{x}_i = [\check{x}_{\neg p} - \kappa(\overline{\boldsymbol{x}}_{\neg p} - \check{x}_{\neg p}),\ \check{x}_{\neg p} + \kappa(\overline{\boldsymbol{x}}_{\neg p} - \check{x}_{\neg p})]$
   endif

4. Form a box $\boldsymbol{x}$ combining $x_{\neg p}$ and $x_p$.
```

ALGORITHM 2. **Form composite boxes**

```
1. Input an integer tag $v$, parameter $p$, reduced box $\boldsymbol{r_b}$, composite
   reduced box $\boldsymbol{c_r}$, inflated box $\boldsymbol{i_b}$, and composite inflated box $\boldsymbol{c_i}$.

2. If integer tag $v$ indicates composite reduced box $\boldsymbol{c_r}$ is input, then
      Do  $1 \le i \le n+1$
         $\underline{\boldsymbol{c_r}} \leftarrow \min(\underline{\boldsymbol{c_r}}, \underline{\boldsymbol{r_b}})$.
         $\overline{\boldsymbol{c_r}} \leftarrow \max(\overline{\boldsymbol{c_r}}, \overline{\boldsymbol{r_b}})$.
      enddo
   else (If integer tag $v$ indicates composite inflated box $\boldsymbol{c_i}$ is input)
      Do  $1 \le i \le n+1$
         $\underline{\boldsymbol{c_i}} \leftarrow \max(\underline{\boldsymbol{c_i}}, \underline{\boldsymbol{i_b}})$.
         $\overline{\boldsymbol{c_i}} \leftarrow \min(\overline{\boldsymbol{c_i}}, \overline{\boldsymbol{i_b}})$.
      enddo
   endif
```

Additional details can be obtained from the author as a technical report.

4. Convergence Properties within Our Algorithm

As outlined in §2.1, we use the local geometry of the function H to determine the widths of the non-parameter coordinates $\boldsymbol{x}_{\neg p}$, constructing the coordinates $\boldsymbol{x}_i$, $1 \le i \le n+1$ with the box-setting algorithm, Algorithm 1. This algorithm then proves existence and uniqueness by verifying:

$$\tilde{\boldsymbol{x}}_{\neg p} \subseteq \mathrm{int}(\boldsymbol{x}_{\neg p}).$$

We now analyze when the above inclusion occurs, and show that, in the absence of singularities, our box-setting algorithm and parameter step control ensure that this condition always occurs. To prove the convergence of $\mathrm{int}(\boldsymbol{x}_{\neg p}) \to \tilde{\boldsymbol{x}}_{\neg p}$ (by repeated applications of the interval Gauss-Seidel method), we need a definition and an assumption.

DEFINITION 1. *If $\tilde{\boldsymbol{x}}_i \subseteq int(\boldsymbol{x}_i)$, say we have inclusion for the i-th coordinate.*

DEFINITION 2. *Let p be a parameter coordinate of an $(n+1)$-dimensional box $\boldsymbol{x}_i$ for $i \le p \le n+1$. Then the shape vector in parameter direction γ is defined as $\boldsymbol{x}_p - x_p$, and the shape vector in non-parameter direction η is defined as $\boldsymbol{x}_{\neg p} - x_{\neg p}$.*

ASSUMPTION 1. *Let $\boldsymbol{x}$ be the $(n+1)$-dimensional box. Let $\tilde{\boldsymbol{x}}$ be the image box of $\boldsymbol{x}$ under one step of the interval Gauss-Seidel method, and let p be the parameter coordinate. Assume $\frac{\partial H}{\partial x_{\neg p}}$ is nonsingular in the box $\boldsymbol{x}$, where $\neg p$ means the p-th column is removed.*

THEOREM 2. *Let Assumption 1 hold and assume that the inverse midpoint preconditioner is used in the interval Gauss-Seidel method. Assume that $H(\check{x})$ is continuously differentiable at the point $\check{x}$, an approximation to the most recently computed point on the curve. Assume that $\check{x}$ has been computed accurately enough to ensure that $0 \in H(\boldsymbol{x})$, where $\boldsymbol{x}$ is a box such that $\check{x} \in \boldsymbol{x}$. Finally, assume that $H'(\boldsymbol{x})$ represents a first- or higher-order interval extension of the Jacobi matrix H' over the box $\boldsymbol{x}$. Then, for small enough predictor stepsize s, the interval Gauss-Seidel method must be successful, i.e., the main algorithm will choose the parameter to be the coordinate of the null vector with largest absolute value, then use the multivariate degree one Taylor polynomial of H as in §2.1 above, to find $\overline{x}_{\neg p}$ and construct a box (as in Algorithm 1 above) so that the inclusion:*

$$\tilde{\boldsymbol{x}}_{\neg p} \subseteq \mathrm{int}(\boldsymbol{x}_{\neg p})$$

will be successful.

Proof. Since $\frac{\partial H}{\partial x_{\neg p}}$ is nonsingular in $\boldsymbol{x}$, $\exists$ a unique locally differentiable curve $x_i = x_i(x_p)$, $i = 1, \ldots, p-1, p+1, \ldots, n+1$ defined in some neighborhood $(\check{x}_p, \check{x}_p + \gamma)$ of $\check{x}_p$ (Implicit Function Theorem [3]) where γ is $\boldsymbol{x}_p - x_p$. In the box-setting algorithm (Algorithm 1), the box $\check{\boldsymbol{x}}$ is defined and constructed as:

$$\check{\boldsymbol{x}}_i = \begin{cases} \boldsymbol{x}_i & \text{if } i = p, \\ \check{x}_i & \text{otherwise.} \end{cases}$$

Thus, $\check{x} \in \check{\boldsymbol{x}}$. Since $0 \in \boldsymbol{H}(\check{x})$ is assumed, $0 \in Y_i \boldsymbol{H}(\boldsymbol{x}_0)$, where $x = (z, \lambda)$ is the inverse midpoint preconditioner matrix defined as $Y = [H'_{\neg p}(\check{x})]^{-1}$. The box-setting algorithm explains how we calculate $\boldsymbol{x}_{\neg p}$. The algorithm sets up a box such that $\check{x}_{\neg p} \in \boldsymbol{x}_{\neg p}$. Then interval Gauss-Seidel iteration finds $\tilde{\boldsymbol{x}}_i$:

$$\tilde{\boldsymbol{x}}_i = \check{x}_{\neg p} - \Big[Y_i \boldsymbol{H}(\check{x}) + \sum_{\substack{i=1 \\ i \neq p}}^{n+1} (Y_i \boldsymbol{H}'_j)(\boldsymbol{x}_j - \check{x}_j) \Big] \Big/ Y_i \boldsymbol{H}'_i. \tag{4}$$

Note that $Y_i \boldsymbol{H}_i$ does not contain zero for small enough $\boldsymbol{x}_{\neg p}$. Let $[\underline{a}, \overline{a}]$ be the numerator in 4 and let $[\underline{b}, \overline{b}]$ be the denominator. Then, since $\boldsymbol{H'}$ is a first or higher-order interval extension of H and since Y_i is the i-th row of the inverse of the midpoint matrix of $\boldsymbol{H'}$, the widths of each component of $\boldsymbol{H}'_i$ tend to zero as the widths of the component of $\boldsymbol{x}$ tend to zero, and moreover, for sufficiently narrow $\boldsymbol{x}$, $\underline{b} > 0$, and, in particular, for sufficiently narrow $\boldsymbol{x}$, $\underline{b} > 1/2$.

Thus, for all $\boldsymbol{x}$ with $w(\boldsymbol{x})$ sufficiently small, we have

$$w(\tilde{\boldsymbol{x}}_i - \check{x}_{\neg p}) = \frac{1}{\underline{b}} w([\underline{a}, \overline{a}]).$$

Thus, for all sufficiently small predictor step size s and taking account of the fact that, due to our construction, the widths of $\boldsymbol{x}_{\neg p}$ get small as the width of the predictor step size s gets small, we obtain

$$w(\tilde{\boldsymbol{x}}_i - \check{x}_{\neg p}) \leq 2w([\underline{a}, \overline{a}]),$$

that is,

$$w(\tilde{\boldsymbol{x}}_i - \check{x}_{\neg p}) \leq 2w\Big[Y\boldsymbol{H}(\check{x}) + \sum_{\substack{i=1 \\ i \neq p}}^{n+1} (Y_i \boldsymbol{H}'_j)(\boldsymbol{x}_{\neg p} - \check{x}_{\neg p}) \Big].$$

The mean value theorem implies that $\boldsymbol{H}'(\boldsymbol{x}) \subset H'(\check{x}) + a_1 (1)_{n \times (n+1)} (\boldsymbol{x} - \check{x})$ for some positive number a_1, where $(1)_{n \times (n+1)}$ is the $n \times (n+1)$ matrix with all components equal to 1. Furthermore, since H is continuously differentiable, H is Lipschitz, which implies $\boldsymbol{H}(\check{x}) \subset a_2 (1)_{n \times 1} (\boldsymbol{x}_p -$

$\check{x}_p)$ for some positive number a_2, where $(1)_{n\times 1}$ stands for the $n \times 1$ matrix with all components equal to 1. Then

$$\begin{aligned}
w(\tilde{\boldsymbol{x}}_i - \check{x}_i) &\leq 2w(Y_i a_2 (1)_{n\times 1}(\boldsymbol{x}_p - \check{x}_p) + \\
&\quad + 2w(\sum_{\substack{i=1\\ i\neq p}}^{n+1} [Y_i(H'(\check{x}) + a_1(1)_{n\times(n+1)}(\boldsymbol{x}-\check{x})]_{\neg p}(\boldsymbol{x}_{\neg p} - \check{x}_{\neg p})) \\
&\leq 2w(Y_i a_2 (1)_{n\times 1}(\boldsymbol{x}_p - \check{x}_p) + \\
&\quad + 2w(\sum_{\substack{i=1\\ i\neq p}}^{n+1} [Y_i a_1 (1)_{n\times(n+1)}(\boldsymbol{x}-\check{x})]_{\neg p}(\boldsymbol{x}_{\neg p} - \check{x}_{\neg p})) \\
&\leq a_3(s + \max[w(\boldsymbol{x}_{\neg p} - \check{x}_{\neg p})]^2)
\end{aligned}$$

where a_3 depends on a_1, a_2 and Y_i, and where s is a predictor stepsize. Then by the box-setting algorithm, we calculate $w(\boldsymbol{x}_{\neg p} - \check{x}_{\neg p})$ as:

$$\begin{aligned}
w(\boldsymbol{x}_{\neg p} - \check{x}_{\neg p}) &= w(\check{x}_{\neg p} + \kappa(\overline{\boldsymbol{x}}_{\neg p} - \check{x}_{\neg p}) - \check{x}_{\neg p} + \kappa(\overline{\boldsymbol{x}}_{\neg p} - \check{x}_{\neg p})) \\
&= w(2\kappa(\overline{\boldsymbol{x}}_{\neg p} - \check{x}_{\neg p})),
\end{aligned}$$

where κ is any positive integer greater than 1. Thus we are increasing $w(\boldsymbol{x}_{\neg p} - \check{x}_{\neg p})$ by multiplying by κ. Let $w(\boldsymbol{x}_{\neg p} - \check{x}_{\neg p}) = \kappa\eta_{max}$ where η is the shape vector for the non-parametric directions (Definition 2). Then

$$\kappa\eta_{min} \leq w(\boldsymbol{x}_{\neg p} - \check{x}_{\neg p}) = \kappa\eta_{max}. \tag{5}$$

Thus

$$\begin{aligned}
w(\tilde{\boldsymbol{x}}_i - \check{x}_i) &\leq a_3(\kappa^2{\eta_{max}}^2 + \gamma) \\
&\leq \frac{a_3(\kappa^2{\eta_{max}}^2 + \gamma)\kappa\eta_{min}}{\kappa\eta_{min}} \\
&\leq \frac{a_3(\kappa^2{\eta_{max}}^2 + \gamma)w(\boldsymbol{x}_i - \check{x}_i)}{\kappa\eta_{min}}
\end{aligned} \tag{6}$$

Let $q = \max\{\mid \check{x} - \underline{\boldsymbol{x}}_i \mid, \mid \check{x} - \overline{\boldsymbol{x}}_i \mid\}$. By the box-setting algorithm (Algorithm 1) we get:

$$q = \kappa\eta_{max}/2,$$

then

$$\frac{w(\boldsymbol{x}_i - \check{x}_i)}{q} \leq \frac{\kappa\eta_{max}}{\frac{\kappa\eta_{max}}{2}}$$

$$
\begin{aligned}
&\le \frac{2\kappa\eta_{max}}{\kappa\eta_{max}}\\
&\le 2.
\end{aligned}
$$

In inequality 6, we need that $\frac{a_3(\kappa^2{\eta_{max}}^2+\gamma)}{\kappa\eta_{min}}$ must approach 0 as the width $w(\boldsymbol{x}_i - \check{x}_i)$ approaches 0. Because of the inequality 5, $w(\tilde{\boldsymbol{x}}_i - \check{x}) \to 0$ means $\kappa\eta_{max} \to 0$ and $\kappa\eta_{min} \to 0$. In other words, $w(\tilde{\boldsymbol{x}}_i - \check{x}) \to 0$ means $\eta_{max} \to 0$ and $\eta_{min} \to 0$.

Then as $\eta_{max} \to 0$ and $\eta_{min} \to 0$,

$$
\begin{aligned}
&\lim_{\eta_{min}\to 0}\Big(\lim_{\eta_{max}\to 0} \frac{a_3(\kappa^2{\eta_{max}}^2+\gamma)}{\kappa\eta_{min}}\Big)\\
=&\lim_{\eta_{min}\to 0}\Big(\lim_{\eta_{max}\to 0} \frac{a_3\kappa^2{\eta_{max}}^2}{\kappa\eta_{min}} + \frac{a_3\gamma}{\kappa\eta_{min}}\Big)\\
=&\lim_{\eta_{min}\to 0}\Big(\lim_{\eta_{max}\to 0} a_3\kappa\frac{({\eta_{max}}^2 - {\eta_{min}}^2 + {\eta_{min}}^2)}{\eta_{min}} + \frac{a_3\gamma}{\kappa\eta_{min}}\Big)\\
=&\lim_{\eta_{min}\to 0}\Big(\lim_{\eta_{max}\to 0} a_3\kappa\frac{(\eta_{max}+\eta_{min})(\eta_{max}-\eta_{min}) + {\eta_{min}}^2}{\eta_{min}} + \frac{a_3\gamma}{\kappa\eta_{min}}\Big)\\
=&\lim_{\eta_{min}\to 0}\Big(\lim_{\eta_{max}\to 0} a_3\kappa\left(\Big(\frac{\frac{\eta_{min}}{\eta_{min}} + \frac{\eta_{max}}{\eta_{min}}}{\frac{\eta_{min}}{\eta_{min}}}\Big)(\eta_{max}-\eta_{min}) + \eta_{min}\right) + \frac{a_3\gamma}{\kappa\eta_{min}}\Big)\\
=&\lim_{\eta_{min}\to 0}\Big(\lim_{\eta_{max}\to 0} a_3\kappa(-\eta_{min}+\eta_{min}) + \frac{a_3\gamma}{\kappa\eta_{min}}\Big)\\
=&\lim_{\eta_{min}\to 0}\Big(\lim_{\eta_{max}\to 0} \frac{a_3\gamma}{\kappa\eta_{min}}\Big) \qquad (7)
\end{aligned}
$$

One thing to notice is the width of the parameter coordinate $w(\boldsymbol{x}_p - x_p) \to 0$, as $\eta_{max} \to 0$ because of the fact that η_{max} is constructed in terms of γ. This can be explained from Equation 7. We now recall from §2.1 that

$$H'_{\neg p}\overline{x}_{\neg p} = H'_{\neg p}(\check{x})_{\neg p} - H(\check{x}) - H'(:,p)(\overline{x}_p - \check{x}_p) \qquad (8)$$

Interchanging the terms and collecting terms in Equation 8 gives

$$H'_{\neg p}(\overline{x}_{\neg p} - \check{x}_{\neg p}) = -H(\check{x}) - H'(:,p)(\overline{x}_p - \check{x}_p) \qquad (9)$$

As $\overline{x}_{\neg p} \to \check{x}_{\neg p}$, the right hand side of Equation 9 becomes 0, since $H(\check{x})$ is approximately 0 at the approximate solution point $\check{x}$. Since the Jacobi matrix $H'_{\neg p}$ is nonsingular, both sides of equation 9 may be multiplied by $\left(H'_{\neg p}\right)^{-1}$, showing that $\overline{x}_{\neg p} \to \check{x}_{\neg p}$. Thus, the width of $\gamma = \boldsymbol{x}_p - \check{x}_p$ goes to 0. Now, combining Equation 7 and the above, we obtain

$$\lim_{\eta_{min}\to 0}\Big(\lim_{\eta_{max}\to 0}\Big(\lim_{\gamma\to 0}\frac{a_3\gamma}{\kappa\eta_{min}}\Big)\Big) = \lim_{\eta_{min}\to 0}\Big(\lim_{\eta_{max}\to 0}\Big(\lim_{\gamma\to 0}\frac{a_3}{\kappa}\Big)\Big).$$

This leads to a condition under which the width of the image of the Gauss-Seidel operator becomes small: For $\gamma > 0,\ \eta_{max} > 0,\ \kappa > 0,\ a_3 > 0$, we must have

$$\frac{a_3}{\kappa} \le \frac{1}{2}, \quad \text{i.e.,} \quad a_3 \le \frac{\kappa}{2}. \tag{10}$$

If the above condition (10) is satisfied, then

$$w(\tilde{\boldsymbol{x}}_i - \check{x}_i) \le q.$$

The fact that 0 is contained in the numerator of equation 4 leads to the fact that $\check{x}_i$ is in both $\boldsymbol{x}_i$ and $\tilde{\boldsymbol{x}}_i$, and a_3 is independent of s. Thus if γ is so small that it makes the condition above (10) valid, then

$$\tilde{\boldsymbol{x}}_i \subseteq \operatorname{int}(\boldsymbol{x}_i) \quad \text{for all} \quad i \neq p.$$

5. Numerical Results

We tested the hybrid interval marching / branch and bound method described in the previous sections on several functions. In all cases, it succeeded. We tried the following functions: Brown's almost linear curve [1], the Layne Watson exponential cosine curve [1], and some other implicit functions [5].

We implemented the method in Fortran 90. For interval computations, a computer software package which can calculate interval arithmetic is essential. For these interval computations, we used INTLIB [26] and INTERVAL_ARITHMETIC [27]. For floating point matrix computations in our program, the LINPACK routines (DSVDC, DGEFA, and DGESL etc.) are utilized. The computer system we used for our research was a Sun Ultra Sparc 1 140 MHZ with 64 MB memory.

Each test function tried in our experiments turned out to be a single curve except for the parametric equation 16 [5], which is composed of six separate curves in a given range ([-1.1,1.1],[-1.1,1.1]). In this function, the first stepsize of each curve following $s^{(1)}$, except for the initial input stepsize s_0, is the cumulative average of stepsizes s from the previous curve following.

The details of numerical results for the functions we have tried are presented below.

Table I. Iteration data for equation 11

n	*number of steps*	*average stepsize*	*number of complement boxes*
2	201	2.5314×10^{-2}	43

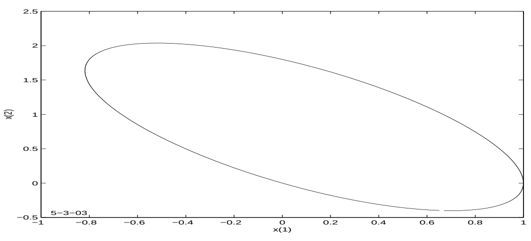

Figure 8. A parametric curve for equation 11 in (x_1, x_2)–space

5.1. Numerical Results on 2×3 dimensional parametric equations

We tried a 2×3 dimensional system of parametric equations we devised. The equation is

$$H(x_i) = \begin{pmatrix} x_1^2 + (\frac{1}{9})x_2^2 + x_3^2 - 1 \\ x_1^2 + x_2^2 + x_3^2 - 1 \end{pmatrix} = 0. \tag{11}$$

The Jacobi matrix of Equation 11 is

$$H'(x_i) = \begin{pmatrix} 2x_1 & (\frac{2}{9})x_2 & 2x_3 \\ 1 & 1 & 1 \end{pmatrix} \tag{12}$$

The results are plotted in Table I. Figure 8 shows a plot of the curve in (x_1, x_2)–space, while Figure 9 shows the curve in (x_1, x_3)–space.

After *all* the complement boxes $\boldsymbol{c_m}$, 43 of them, are constructed, our search algorithm rejects *all* the complement boxes, confirming that there are no curves in the list of complement boxes $\boldsymbol{l_c}$.

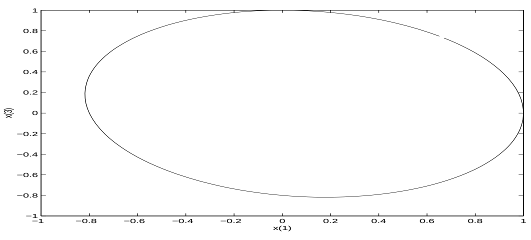

Figure 9. A parametric curve for equation 11 in (x_1, x_3)–space

Table II. Iteration data for Brown's almost linear curve in different dimensions

n	*number of steps*	*average stepsize*	*number of complement boxes*
2	128	1.9385×10^{-2}	28
5	1091	5.3213×10^{-3}	190

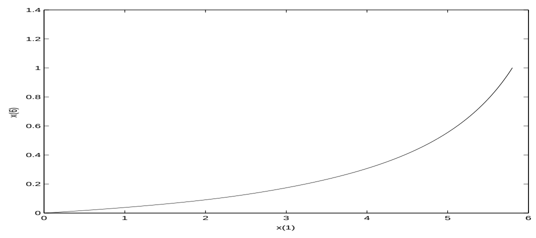

Figure 10. Brown's almost linear curve in 5 × 6 dimensions

5.2. Numerical Results on Brown's almost linear curve

Brown's almost linear curve is defined as

$$f_i(X) = x_i + x_{n+1}\left(\sum_{1\le j\le n} x_j - n - 1\right), \qquad 1 \le i \le n-1 \tag{13}$$

and

$$f_n(X) = (1 - x_{n+1})x_n + x_{n+1}\left(\prod_{1\le j\le n} x_j - 1\right). \tag{14}$$

We tried different dimensional instances of Brown's almost linear curve. The results in Table II represent marching and search over the box with x_{n+1} bounded between 0 and 1 and with $x_{\neg(n+1)}$ allowed to vary. With any initial points, our program will find a feasible starting point. The results for dimension n=2 turned out to be compatible with the results shown in [1]. Our results confirm the results from [1], in that our results show that the stepsize does not increase with dimension.

Figure 10 shows the points on the Brown's almost linear curve plotted x_1 versus x_6, where x_6 is x_{n+1}, and x_6 changes from 0 to 1.

After *all* the complement boxes $\boldsymbol{c_m}$, are constructed, our search algorithm rejects *all* the complement boxes, confirming that there is no curve in the list of complement boxes $\boldsymbol{l_c}$.

Since our program is intended mainly for graphical computations, we did not try $n > 5$.

Table III. Layne Watson exponential cosine curve in 2×3 dimensions

n	*number of steps*	*average stepsize*	*number of complement boxes*
2	318	3.5834×10^{-3}	24

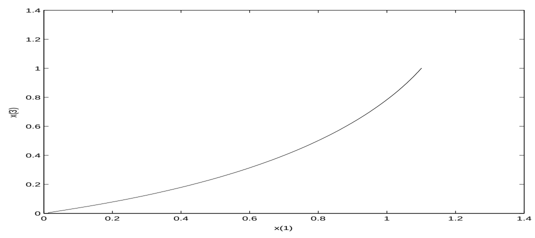

Figure 11. Layne Watson exponential cosine curve in 2×3 dimensions

5.3. Numerical Results on the Layne Watson exponential cosine curve

The Layne Watson exponential cosine curve is defined as

$$f_i(X) = x_i - x_{n+1} e^{\cos i (\sum_{1 \le j \le n} x_j)}, \qquad 1 \le i \le n \tag{15}$$

We tried the 2×3 dimensional instance of the Layne Watson exponential cosine curve. The results in Table III represent marching and search over the box with x_{n+1} bounded between 0 and 1 and with $x_{\neg(n+1)}$ allowed to vary. With any initial points, our program will find a feasible starting point. Our results for dimension $n = 2$ turned out to be compatible with the results shown in [1]. Figure 11 shows the points on the Layne Watson Exponential Cosine Curve plotted x_1 versus x_3. x_{n+1}, which is, x_3, changes from 0 to 1.

After *all* the complement boxes $\boldsymbol{c_m}$ are constructed, our search algorithm rejects *all* the complement boxes, 24 of them, confirming that there is no curve in the list of complement boxes $\boldsymbol{l_c}$.

5.4. Numerical Results on a 1×2 dimensional parametric equation

We tried a 1×2 dimensional system of parametric equations [5]. The equation is

$$H(x_1, x_2) = x_1^2 + x_2^2 + \cos(2\pi x_1) + \sin(2\pi x_2) + \sin(2\pi x_1^2)\cos(2\pi x_2^2) - 1 \tag{16}$$

Table IV. Iteration data for 6 curves for equation 16

curve	*number of steps*	*starting point x(1)*
1	1442	(0.7966,0.8364)
2	644	(0.1916,-0.4662)
3	102	(0.7001,-0.0573)
4	89	(-0.4885,-1.0343)
5	36	(1.0598,0.7569)
6	38	(-1.0748,0.7761)

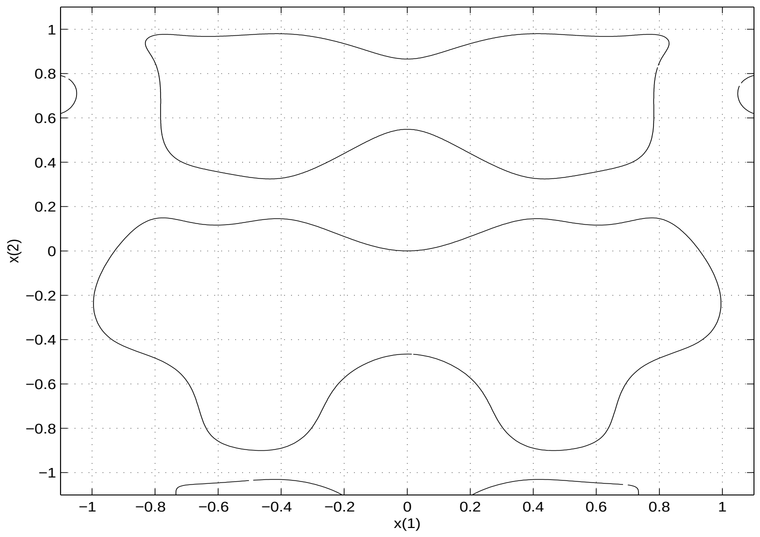

Figure 12. 6 curves of the parametric equation 16

Our curve following algorithm confirms that the equation is composed of six separate curves in a given range box b_0 ([-1.1,1.1],[-1.1,1.1]). The results of the iteration are in Table IV.

Table IV shows how the six curves behave. The average stepsize for the function is 2.41×10^{-3}. Figure 12 shows that the points on the parametric equation 16 plotted x_1 versus x_2. The curves 3 to 6 hit the edge of the range box b_0([-1.1,1.1],[-1.1,1.1]) once, so the algorithm went to the first point, and then followed the curve in the direction opposite to the first direction.

Figure 13 shows the plot of the points of the curve and the complement boxes together on the same graph, after *all* 6 curves are found by our algorithm.

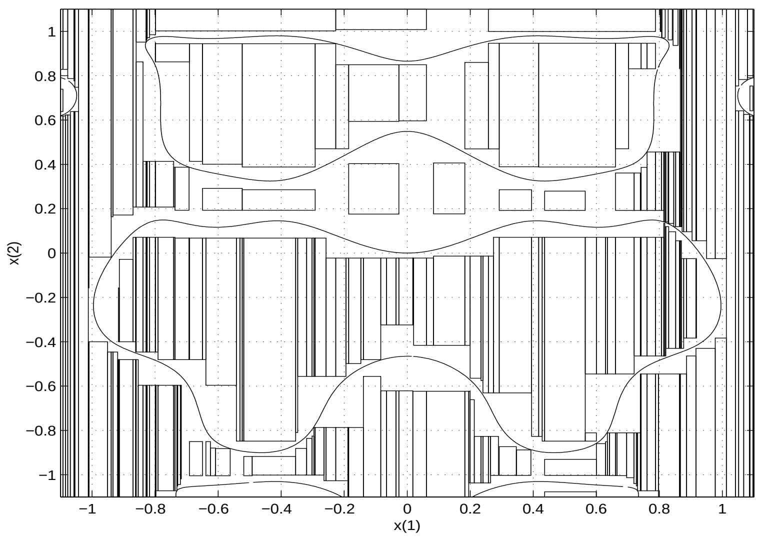

Figure 13. 6 curves and complement boxes for equation 16

Table V. Number of complement boxes after each curve for equation 16

curve	*number of complement boxes*
1	68
2	242
3	241
4	241
5	251
6	263

The CPU time for this particular iteration (for the parametric equation 16) is 50.95 seconds on a Sun Ultra Sparc 1 140 MHZ with 64 MB memory. We suspect that much of this time was spent in printing operations.

Table V lists the number of complement boxes formed after each curve following. Observe that table V shows that some complement boxes $\boldsymbol{c_m}$ are already rejected by our algorithm, once the list of complement boxes $\boldsymbol{l_c}$ for each curve is formed. Figure 14 is the plot of the complement boxes only, after *all* 6 curves are found.

After *all* the complement boxes $\boldsymbol{c_m}$ for *all* 6 curves are constructed, our *global* search algorithm rejects *all* 263 complement boxes, confirming that there is no curve in the list of complement boxes $\boldsymbol{l_c}$.

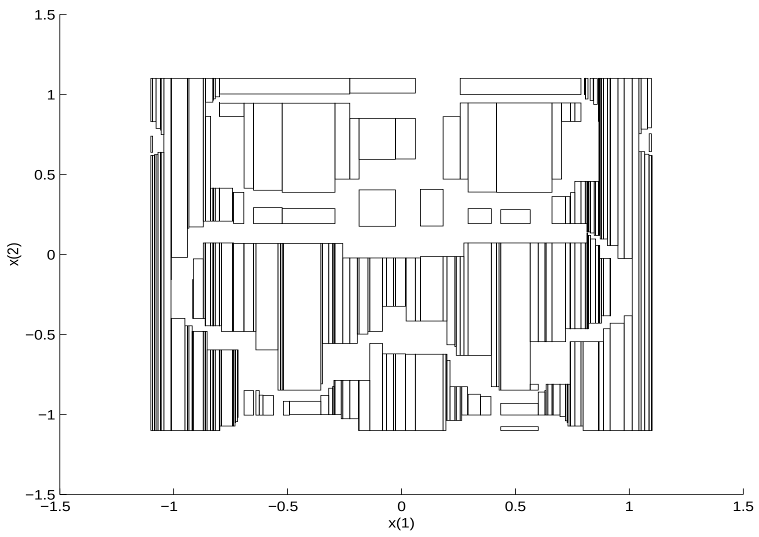

Figure 14. Complement boxes of the implicit curve for equation 16

References

1. R. BAKER KEARFOTT AND ZHAOYUN XING: *An Interval Step Control for Continuation Methods*, joint with Z. Xing, SIAM J. Numer. Anal. 31 (3), pp. 892-914 (1994).
2. R. BAKER KEARFOTT *On Proving Existence of point at which $H(Y) = 0$s in Equality Constrained Optimization Problems*, Mathematical Programming 83 (1), pp. 89-100 (September, 1998).
3. EUGENE L. ALLGOWER and K. GEORG: *Introduction to Numerical Continuation Methods*, Colorado State University, 1990.
4. R. BAKER KEARFOTT: *Rigorous Global Search*, Kluwer, Dordrecht, 1996.
5. JOHN M. SNYDER: *Interval Analysis For Computer Graphics*, Siggraph 92, 26 (2): 121-130, 1992.
6. R. BAKER KEARFOTT AND ZHAOYUN XING: *Rigorous Computation of Surface Patch Intersection Curves*, Preprint, 1993.
7. R. E. MOORE: *Interval Analysis*, Prentice Hall, New Jersey, 1966.
8. ELDON HANSEN: *Global Optimization Using Interval Analysis*, Marcel Dekker, New York, 1992.
9. ARNOLD NEUMAIER: *Interval Methods for Systems of Equations*, Cambridge University Press, 1990.
10. R. B. KEARFOTT: *Interval Computations: Introduction, Uses, and Resources*, Euromath Bulletin, 2(1):95 112, 1996.
11. R. B. KEARFOTT: *Preconditioners for the interval Gauss-Seidel method*, SIAM J. Numer. Anal., 27(3):804-822, June 1990.
12. R. B. KEARFOTT, C. HU, and M. NUVOA:*A Review of Preconditioners for the Interval Gauss-Seidel Method*, Interval Computations, 1(1):59 85, 1991.
13. C. HU:*Optimal Preconditioners for the Interval Newton Method*, PhD Thesis, University of Louisiana at Lafayette, 1990.

14. X. SHI:*Intermediate Expression Preconditioning and Verification for Rigorous Solution of Nonlinear Systems*, PhD Thesis, University of Louisiana at Lafayette, 1995.
15. D. RATZ:*Automatische Ergebnisverifikation bei Globalen Optimierungsproblemen*, PhD Thesis, Universität of Karlsruhe, 1992.
16. G. ALEFELD and J. HERZBERGER: *Introduction to Interval Computations*, Academic Press, New York, 1983.
17. J. GARGANTINI and P. HENRICI: *Circular arithmetic and the determination of polynomial zeros*, Numer. Math., 18:305-320, 1972.
18. L. E. J. BROUWER: *Über Abbildung von Mannigfaltigkeiten*, Math. Ann., 17:97-115, 1912.
19. C. MIRANDA: *Un' osservatione su un teorema di Brouwer.*, Bol, Un, Mat. Ital., Series 2, 2:5-7, 1940.
20. H. BRÖNNIMANN, C. BURNIKEL, S. PION: *Interval Arithmetic Yields Efficient Dynamic Filter for Computational Geometry.*, In Proc. 14th Annu. ACM Sympos. Compu. Geom., pages 165–174, 1998.
21. S. KRISHNAN, and D. MANOCHA: *An Efficient Surface Intersection Algorithm Based on Lower-Dimensional Formulation.*, ACM Transactions on Graphics, Vol. 16, No. 1, January 1997, pages 74–106, 1997.
22. JOHN M. SNYDER: *Interval Methods for Multi-Point Collisions between Time-Dependent Curved Surfaces.*, Computer Graphics(SIGGRAPH'93 Proceedings), volumn 27, pages 321–334, 1993.
23. S. P. MUDUR, and, P. A. KOPARKAR: *Interval Methods for Processing Geometric Objects.*, IEEE Comput. Graphics and Appl., 4(2):7-17, 1984.
24. P. SCHRAMM: *Intersection Problems of Parametric Surfaces in CAGD.*, Computing, 53:355-364, 1994.
25. T. W. SEDERBERG and S. R. PARRY:*Comparison of Three Curve Intersection Algorithms.*, Comput. Aided Des., 18(1):58-63, 1986.
26. R. B. KEARFOTT: *INTLIB, a portable FORTRAN 77 interval standard function library.*, ACM Trans. Math. Software, 20(4):447-459, December 1994.
27. R. B. KEARFOTT: *Algorithm 763(Interval Arithmetic), A (Fortran 90) Module for an Interval Data Type.* ACM Trans. Math. Software, 22(4):385-392, December 1996.

Address for Offprints:
Mihye Kim,
103-C, St. Neri, Youngsville, LA, 70592
e-mail: mihyekim@yahoo.com