SIAM J.SIAM J. Numer. Anal.
Vol. 0, No. 0, 1993

AN INTERVAL STEP CONTROL FOR CONTINUATION METHODS

R. BAKER KEARFOTT AND ZHAOYUN XING [†]

Abstract. We present a step control for continuation methods that is deterministic in the sense that (i) it computationally but rigorously verifies that the corrector iteration will converge to a point on the same curve as the previous point (i.e. the predictor / corrector iteration will never jump across paths), and (ii) each predictor step is as large as possible, subject to verification that the curve is unique with the given interval extension. We present performance data and comparisons with an approximate step control method (PITCON version 6.1). A comparison of plots obtained from both step controls reveals that an approximate step control will behave erratically in situations where the interval step control leads to orderly progression along the curve. This is true even if the maximum allowable stepsize for the approximate method is set to be smaller than many of the steps actually taken by the interval algorithm. We also discuss limitations of interval step controls.

Key words. automatic verification, interval iteration, uniqueness, Gauss–Seidel iterations, continuation methods, step control, PITCON.

AMS(MOS) subject classifications. 65G10, 65H10, 65H20

1. Notation and Introduction.

Throughout this paper, we use boldface lowercase letters (such as $\mathbf{x}$) for interval vectors, uppercase letters (such as X) for point vectors, and lowercase letters with subscripts (such as x_i) for scalar variables. An exception will be the symbol $\boldsymbol{\eta}$, which will denote a vector of real numbers $\boldsymbol{\eta}_i$. We will use notation such as $\mathbf{H}(\mathbf{x})$ to denote an interval extension of $H(x)$ over $\mathbf{x} \ni x$.

In this paper, we consider following problem:
Compute a sequence of points on the solution manifold

$$\mathscr{Z} = \{Y \in \mathbb{R}^{n+1} | H(Y) = 0\}, \text{ where } H : \mathbb{R}^{n+1} \to \mathbb{R}^n,$$

with a guarantee that all the points are on the same continuous path. For an introduction to classical methods for following paths in $\mathscr{Z}$, as well as a large number of references, see [1]. So far, most of the literature, such as [3], on following the curve is based on approximate step control. Approximate step control methods are successful and fast when the curve is smooth and isolated, but problems arise when there are many paths near some points. In that case, algorithms based on approximate step control methods may jump from one path to another, as the numerical results in §4 show. Also, if rapid changes in curvature occur along the path, the method based on approximate step control sometimes even erroneously reverses orientation. However, appropriate use of interval analysis gives us a guarantee that the predictor algorithm will not jump from one path to another, or, indeed, jump over different legs of the same path.

We assume that our algorithms start with a point $X_0 \in \mathbb{R}^{n+1}$ on $\mathscr{Z}$ (i.e. $H(X_0) = 0$). If the Jacobian matrix $H'(X)$ is of full rank at X_0, then the tangent vector $B(X_0)$

[†] U.S.L. Box 4-1010, Lafayette, LA 70504-1010 This work was supported in part by the National Science Foundation under Grant CCR-9203730.

can be computed to within length and orientation. This information can then be used to choose an appropriate parameter coordinate. Following [3], we choose the parameter coordinate adaptively, so that curve following is as fast as possible and is successful. That is, we choose i_0 such that

$$|B_{i_0}| = \max_{1 \le i \le n+1} |B_i|.$$

If $\mathrm{sgn}(B_{i_0'}) = -\mathrm{sgn}(B^{(1)}_{i_0'})$, where i_0' and $B^{(1)}$ are the previous parameter number and tangent vector, respectively, we change the orientation of this tangent vector $B(X_0)$. This action rigorously maintains constant orientation, as is proven in Theorem 3.5 below.

We use a real vector $\boldsymbol{\eta}(X_0)$ with positive components as well as the predictor stepsize δ to control the shape of the box[1]. If the curve can be parametrized locally in terms of the parameter coordinate x_{i_0}, then a box $\mathbf{x}$ containing X_0 on one of its faces and further specified by $\boldsymbol{\eta}, \delta$ and B is constructed according to Algorithm 2.3 below. The box $\mathbf{x}$ is chosen to make it likely that a piece of the curve passes through its faces that are perpendicular to the parameter coordinate axis. The algorithm will then rigorously verify this with the interval Gauss–Seidel method, as we now outline.

Given $\mathbf{x}$, define $\mathbf{x}_0$ by

$$\mathbf{x}_{0,i} = \begin{cases} X_{0,i}, & \text{for } i \ne i_0; \\ \mathbf{x}_{i_0}, & \text{for } i = i_0. \end{cases}$$

Here, $\mathbf{x}_0$ has an interval instead of a point in its parameter coordinate (i_0). This is because we wish to prove uniqueness of the solution with respect to the other coordinates, for *every* value $X_{i_0} \in \mathbf{x}_{i_0}$. Thus, the preconditioned "point" function value $\mathbf{k}_i$ must be an interval. We have not seen an alternative in our analysis.

Given the solution bounds $\mathbf{x}_i$, $i \ne i_0$, we will compute new solution bounds $\overline{\mathbf{x}}_i$ by the optimally preconditioned interval Gauss–Seidel method ([4]) as follows. The box $\mathbf{x}$ is constructed to make it likely that the curve passes through the faces perpendicular to the parameter coordinate. Theorems 3.2 and 3.3 below then guarantee that the curve is contained in a suitably constructed box $\mathbf{x}$, and Theorem 3.4 guarantees that, under certain circumstances, Algorithm 2.1 will eventually be successful. We make use of the following notation and definition.

For suitably chosen preconditioning (row) vectors Y_i, define

$$\mathbf{k}_i = Y_i \mathbf{H}(\mathbf{x}_0),$$
$$\mathbf{G}_i = Y_i \mathbf{H}'(\mathbf{x}),$$

and, for suitably chosen preconditoning row vectors Y_i, let

$$\overline{\mathbf{x}}_i = \mathbf{x}_{0,i} - \left[\mathbf{k}_i + \sum_{\substack{j=1 \\ j \ne i}}^{n} \mathbf{G}_{ij}(\mathbf{x}_j - \mathbf{x}_{0,j}) \right] \Big/ \mathbf{G}_{ii},$$

where $\mathbf{G}_{i,j}$ is the j-th entry of $\mathbf{G}_i$. If $\overline{\mathbf{x}}_i \subseteq \mathrm{int}(\mathbf{x}_i)$ for all $i \ne i_0$, where int() denotes the topological interior, then by Theorem 3.1 below, there is a unique curve in the box $\mathbf{x}$ passing through the two faces of $\mathbf{x}$ defined by:

$$x_{i_0} = \inf(\mathbf{x}_{i_0}) \text{ and } x_{i_0} = \sup(\mathbf{x}_{i_0}).$$

[1]i.e. rectangular parallelpiped, representable as an interval vector.

DEFINITION 1.1. If $\overline{\mathbf{x}}_i \subseteq \text{int}(\mathbf{x}_i)$, we say that we **have inclusion** for the i-th component.

If $\overline{\mathbf{x}}_i \not\subseteq \text{int}(\mathbf{x}_i)$, that is, if we do not have inclusion for some component, our step control algorithm will change the shape of the box in an appropriate way, then repeat the above computation.

2. Specific Algorithms.

Our algorithms are based upon heuristics for adjusting both a shape vector $\boldsymbol{\eta}$ and the predictor stepsize δ. The heuristics affect the efficiency of the algorithm, but not its rigor.

Theorems 3.2-3.5 below tell us that, under certain circumstances, the larger $\boldsymbol{\eta}_i$ is, the more likely it is to get inclusion for the i-th component. Also, larger stepsizes δ are desirable for various reasons. Therefore, if inclusion is not signaled in the algorithm, we consider increasing $\boldsymbol{\eta}_i$ before decreasing δ. However, overly large box widths may result in overestimation, which may be reflected by the inability to compute the optimal preconditioner[2] of [4]. For this reason, we decrease $\boldsymbol{\eta}_i$ if preconditioner computation is unsuccessful. After a successful preconditioner computation, we increase $\boldsymbol{\eta}_i$ if we do not have inclusion.

However, if $\boldsymbol{\eta}_i$ has been cut in a previous step because of failure of the preconditioner computation, then, to avoid cycling, $\boldsymbol{\eta}_i$ should not be increased. In this case, if inclusion is not obtained in the i-th coordinate, we decrease the stepsize δ. We use a logical flag $\mathcal{I}_i$ to record whether $\boldsymbol{\eta}_i$ has ever been cut, and we set a flag L_1 to the bound for the number of times δ can be cut after $\boldsymbol{\eta}_i$ has been cut. If, after N_1 adjustments of $\boldsymbol{\eta}_i$ and δ, we still cannot get inclusion, we increase L_1. Another algorithm parameter K is used: if we still cannot get inclusions for all components after KN_1 iterations, we say that the algorithm fails.

After the new point is computed, we set the new stepsize according to the following: An algorithm parameter L_2 is used to detect when caution should be exercised when increasing δ. Normally, we double the stepsize δ. However, if L_1 was increased to the point that $L_1 > L_2$, we increase δ by a smaller amount, say $\delta \leftarrow 1.2\delta$.

Details of the adjustment process for $\boldsymbol{\eta}_i$ and δ are presented in Algorithm 2.4, executed in Step 5 of the following overall algorithm.

ALGORITHM 2.1. (Overall algorithm)

```
1. Input the known point $X_0$ on the curve, the functions $H$ and $H'$, the stepsize $\delta$
   and parameters $\delta_{\max}$, $\eta_{\min}$, $\eta_{\max}$, $\eta_1$, $K$, $n$, $N_1$, $L_1$, and $L_2$. Also input an initial
   tangent vector $B^{(1)}$ and parameter coordinate number $i_0'$, to be used to orient the
   tangent vector on the first step.
2. For all $1 \le i \le n+1$, set
      $\mathcal{I}_i \leftarrow 0$,
      $\boldsymbol{\eta}_i \leftarrow \eta_1$.
3. (Compute the tangent vector $B(X_0)$ and parameter $i_0$, using previous values for
   orientation)
         Execute Algorithm 2.2.
         Store $B$ and $i_0$:
               $B^{(1)} \leftarrow B$,
               $i_0' \leftarrow i_0$.
```

[2]If inclusion can be obtained, then it is possible to compute an optimal preconditioner.

```
4. $i \leftarrow 1$,
   $k \leftarrow 0$.
5. **If** $k > K$, **then**
         *Print "This algorithm fails," then stop.*
   **else**
         **If** $i \le N_1$, **then**
           a) (Construct the box $\mathbf{x}$.) *Execute Algorithm 2.3.*
           b) (Do a sweep of Gauss–Seidel iteration and possibly alter $\boldsymbol{\eta}$ and $\delta$.) *Execute Algorithm 2.4. Return if inclusion is obtained for all components of* $\mathbf{x}$.
           c) **If** *there is a component for which inclusion is not obtained,* **then**
                α) $i \leftarrow i + 1$.
                β) *Goto Step a).*
              **else**
                   *Continue.*
              **endif**
         **else**
           d) $L_1 \leftarrow L_1 + 1$,
              $i \leftarrow 1$,
              $k \leftarrow k + 1$.
           e) *Goto Step a).*
         **endif**
   **endif**
6. *Use the classical multivariate Newton method to find an approximation to the new point $X_1$ on the curve. That is, find an approximation to the point of intersection of the curve with the hyperplane $x_{i_0} = \sup(\mathbf{x}_{i_0})$ if $B_{i_0} > 0$ and $x_{i_0} = \inf(\mathbf{x}_{i_0})$ if $B_{i_0} \le 0$.*
7. **If** *the stopping criterion is satisfied,* **then**
         *Stop.*
   **else**
     a) **If** $L_1 > L_2$ **then**
              $\delta \leftarrow 1.2\delta$;
        **else**
              $\delta \leftarrow 2\delta$.
        **endif**
     b) *If* $\delta \ge \delta_{\max}, \delta \leftarrow \delta_{\max}$.
     c) $L_1 \leftarrow L_1 - 1$.
     d) *Goto Step 2.*
   **endif**
```

In our experiments in §4, we use the following LU-decomposition based technique to compute the tangent vector.

Algorithm 2.2. (Finding the tangent vector)

```
1. Input the present point $X_0$ on the curve, the tangent vector $B^{(1)}$ corresponding to the point on the curve computed immediately before $X_0$ and the parameter coordinate $i'_0$ corresponding to this previous point.
2. Compute the $n \times (n+1)$ Jacobian matrix $H'(X_0)$ of $H$ at the point $X_0$.
3. Initially set $i_{00} = n + 1$, then delete the $i_{00}$-th column of $H'(X_0)$ to obtain an $n \times n$ matrix ${H_0}'(X_0)$.
```

4. *Obtain an LU factorization and estimate of the condition number for the matrix $H_0{}'(X_0)$ from* Step 3 *(e.g. with the LINPACK routine DGECO). If the estimate for the condition number is higher than a given tolerance (e.g. 10^{-4}), locate the column of U with the approximate 0 on the diagonal, and label its index i_{00}.*
5. *Delete the i_{00}-th column from the original Jacobian matrix $H'(X_0)$ to produce a new $H_0{}'(X_0)$. Solve the resulting linear system $H'(X_0)Y = -H'_{:,i_{00}}$ for Y, where $H'_{:,i_{00}}$ represents the i_{00}-th column of $H'(X_0)$.*
6. *Construct the tangent vector B by first setting $B_{i_{00}} = 1$, then setting the other components of B equal to corresponding components of Y.*
7. *Find i_0 such that $|B_{i_0}| = \max_{1 \le i \le n+1}\{|B_i|\}$.*
8. (Set the orientation) *If $B_{i_0'} B^{(1)}_{i_0'} < 0$, set $B \leftarrow -B$.*
9. (Normalize) *Set $B \leftarrow B/|B_{i_0}|$.*

In the following algorithm, a box is constructed which contains the point X_0 in the interior of one of its faces, which contains the line segment from X_0 corresponding to linearization of the path, which contains $X_{0,i}$ in the interior of its i-th coordinate for $i \ne i_0$, which takes account of tangent information, and whose widths heuristically account for nonlinearities and overestimation in the interval extensions through the parameter vector $\boldsymbol{\eta}$. When $\boldsymbol{\eta}$ is small and the component B_{i_0} is relatively small, the construction makes $X_{0,i}$ into the midpoint of the coordinate interval $\mathbf{x}_i$. We believe this to make it more likely to have inclusion.

Algorithm 2.3. (Use the known point X_0 and tangent $B(X_0)$ to construct the box shape.)

```
1. Input $\delta$, $\boldsymbol{\eta}$, $X_0$ and $B(X_0)$.
2. If $i = i_0$, then
       set
           $\mathbf{x}_i = \begin{cases} [X_{0,i}, X_{0,i} + B_{i_0}\delta] & \text{if } B_{i_0} \ge 0 \text{ and} \\ [X_{0,i} - B_{i_0}\delta, X_{0,i}] & \text{if } B_{i_0} < 0. \end{cases}$
   else
       set
           $p_{i,1} = X_{0,i} + B_i\delta - \boldsymbol{\eta}_i\delta$
           $p_{i,2} = X_{0,i} + B_i\delta + \boldsymbol{\eta}_i\delta$
       and
           $\mathbf{x}_i = \begin{cases} [2X_{0,i} - p_{i,2}, p_{i,2}] & \text{if } p_{i,1} > X_{0,i}, \\ [p_{i,1}, 2X_{0,i} - p_{i,1}] & \text{if } p_{i,2} < X_{0,i}, \text{ and} \\ [p_{i,1}, p_{i,2}) & X_{0,i} \in (p_{i,1}, p_{i,2}]. \end{cases}$
   end if
```

See figure 2.1 for a graphical illustration of the above construction.

The following algorithm is similar to Algorithm 1.2 in [4] and elsewhere, except that we use information about the preconditioner computation to determine how to adjust the box dimensions.

Algorithm 2.4. (Do one sweep of interval Gauss–Seidel iteration to determine whether there is a unique curve in the constructed box. Also adjust the box shape as appropriate.)

1. *Input the previously computed point on the curve X_0, the constructed box $\mathbf{x}$, the stepsize δ, $\delta_{\min}$, $\boldsymbol{\eta}$, $\eta_{\min}$, $\eta_{\max}$, L_1, and i_1. (Here, i_1 is the number of times δ has been cut due to failure to compute a preconditioner, when checking the j-th component and $\mathcal{I}_j = 1$.) Also input the flag vector $\mathcal{I}$ whose components indicate*

FIGURE 2.1. CONSTRUCTION OF $\mathbf{x}_i$, $i \neq i_0$ IN THE CASE $X_{0,i} \in (p_{i,1}, p_{i,2})$.

```
     whether preconditioner computation for the corresponding variable was successful
     in the previous invocation of this algorithm.
  2. Compute $\mathbf{x}_0$, where
        $\mathbf{x}_{0,i} = \begin{cases} \mathbf{x}_i & \text{if} \quad i = i_0 \\ X_{0,i} & \text{otherwise.} \end{cases}$
     While $i \neq i_0$ and $1 \leq i \leq n+1$ Do Step 3 to Step 7.
  3. Compute an interval extension $\mathbf{H}'(\mathbf{x})$ to the Jacobian matrix and compute the
     interval function value $\mathbf{H}(\mathbf{x}_0)$.
  4. Compute the preconditioner row $Y_i$ as in [4] or [5].
     If preconditioner computation was not successful then
       a) $\mathcal{I}_i \leftarrow 1$,
       b) (Either decrease all components of $\boldsymbol{\eta}$ or decrease $\delta$.)
          α) $\boldsymbol{\eta}_j \leftarrow \boldsymbol{\eta}_j/8$, for all $1 \leq j \leq n+1$.
          β) If $\boldsymbol{\eta}_j \leq \eta_{\min}$, for some $j$, then
                 $\boldsymbol{\eta}_j \leftarrow \eta_{\min}$,
             endif
          γ) If $\boldsymbol{\eta}_j = \eta_{\min}$ for all $1 \leq j \leq n+1$ then
                 $\delta \leftarrow \delta/2$.
             endif
       c) Return $\boldsymbol{\eta}$ and $\delta$.
     else Set
          $\mathbf{k}_i = Y_i \mathbf{H}(\mathbf{x}_0)$,
          $\mathbf{G}_i = Y_i \mathbf{H}(\mathbf{x})$, and
          $\overline{\mathbf{x}}_i = \mathbf{x}_{0,i} - \left[ \mathbf{k}_i + \sum_{\substack{j=1 \\ j \neq i, i_0}}^{n+1} \mathbf{G}_{i,j}(\mathbf{x}_j - \mathbf{x}_{0,j}) \right] \Big/ \mathbf{G}_{ii}$.
```

```
    **If** $\overline{\mathbf{x}}_i \subseteq \mathrm{int}(\mathbf{x}_i)$, **then**
      a) key $\leftarrow 1$,
      b) $i \leftarrow i+1$,
      c) Return to the beginning of Step 4.
    **else**
      d) key $\leftarrow 2$,
      f) **If** $\mathcal{I}_i = 1$, **then**
             **If** $i_1 < L_1$, **then**
                 $\delta \leftarrow \delta/2$,
                 $i_1 \leftarrow i_1 + 1$.
             **else**
                 $\boldsymbol{\eta}_i \leftarrow 2\boldsymbol{\eta}_i$.
                 **If** $\boldsymbol{\eta}_i > \eta_{\max}, \boldsymbol{\eta}_i \leftarrow \eta_{\max}$,
                 $\delta \leftarrow \delta/2$.
             **endif**
         **else**
             **If** $\boldsymbol{\eta}_i > \eta_{\max}$ **then**
                 $\boldsymbol{\eta}_i \leftarrow \eta_{\max}$,
                 $\delta \leftarrow \delta/2$.
             **endif**
         **endif**
      g) Return $\boldsymbol{\eta}$ and $\delta$.
    **endif**
**endif**
```

3. Mathematical Theorems.

ASSUMPTION 1. Assume $H : \mathbb{R}^{n+1} \to \mathbb{R}^n$ has continuous derivatives. Assume x_{i_0} is the parameter coordinate chosen in Algorithm 2.2. For simplicity but without loss of generality, assume throughout this section that the parameter coordinate $i_0 = 1$. Let $\mathbf{x}$ denote the $(n+1)$-dimensional box and let $\mathbf{x}_{-1}$ denote the n-dimensional box obtained from $\mathbf{x}$ by omitting the first coordinate. Similarly, assume $\tilde{\mathbf{x}}_{-1}$ is obtained from $\tilde{\mathbf{x}}$, the image box of $\mathbf{x}$ under one step of the Gauss–Seidel iteration described in Algorithm 2.4, by omitting the first coordinate of $\tilde{\mathbf{x}}$. Also assume that $\dfrac{\partial H}{\partial X_{-1}}$ is nonsingular in the box $\mathbf{x}$, where $\dfrac{\partial H}{\partial X_{-i}}$ the Jacobian matrix of $\mathbf{H}$ with i-th column removed.

THEOREM 3.1. *Suppose Assumption 1 is true; also assume that*

$$\tilde{\mathbf{x}}_{-1} \subseteq \mathrm{int}(\mathbf{x}_{-1}),$$

where $\mathrm{int}(\mathbf{x})$ *represents the interior of the set* $\mathbf{x}$. *Then there exists a unique smooth path passing through the two faces of the box* $\mathbf{x}$ *defined by*

$$\{X \in \mathbf{x} \mid x_1 = \sup\{\mathbf{x}_1\}\} \tag{1}$$

and

$$\{X \in \mathbf{x} \mid x_1 = \inf\{\mathbf{x}_1\}\}\,. \tag{2}$$

Proof. For any fixed $x_1 \in \mathbf{x}_1$ on the hyperplane $X_1 = x_1$, execute Algorithm 2.4 (a sweep of the preconditioned interval Gauss–Seidel method) with $\mathbf{x}' = x_1 \times \mathbf{x}_{-1}$ in place of $\mathbf{x}$ and $\mathbf{x}'_0$ in place of $\mathbf{x}_0$, where

$$\mathbf{x}'_{0,i} = \begin{cases} x_1 & \text{when } i = 1, \\ \mathbf{x}_{0,i} & \text{otherwise.} \end{cases}$$

Since $x_1 \in \mathbf{x}_1$, $\mathbf{x}'_0 \subset \mathbf{x}_0$. Since $\mathbf{x}' \subset \mathbf{x}$, inclusion monotonicity implies for all $i \neq 1$, we have

$$\tilde{\mathbf{x}}'_i \subset \tilde{\mathbf{x}}_i \subset \text{int}(\mathbf{x}_i).$$

If we define $\tilde{\mathbf{x}}'_{-1}$ in the same way as $\tilde{\mathbf{x}}_{-1}$, with $\mathbf{x}_{-1} = \mathbf{x}'_{-1}$, then

$$\tilde{\mathbf{x}}'_{-1} \subset \text{int}(\mathbf{x}'_{-1}).$$

According to Theorem 5.1.8 in [7], there is a unique solution to

$$\begin{cases} H(X) & = 0 \\ X_1 & = x_1 \end{cases}$$

for any *fixed* value $x_1 \in \mathbf{x}_1$. Denote this point by

$$(x_1, x_2(x_1), x_3(x_1), \ldots, x_{n+1}(x_1))^T.$$

Define a space $\mathcal{Y}$ by

$$\mathcal{Y} = \{(x_1, x_2(x_1), \ldots, x_{n+1}(x_1))^T \mid x_1 \in \mathbf{x}_1\},$$

where $\pi((x_1, x_2(x_1), x_3(x_1), \ldots, x_{n+1}(x_1))^T) = x_1$. The space $\mathcal{Y}$ has a topology defined by the subspace topology inherited from $\mathbb{R}^n$. Define a map

$$\pi : \mathcal{Y} \to \mathbf{x}_1,$$

where $\mathbf{x}_1$ is viewed topologically as a subspace of $\mathbb{R}^1$. A projection π is well defined. Since, by Assumption 1, $\dfrac{\partial H}{\partial X_{-1}}$ is nonsingular in $\mathbf{x}$, the implicit function theorem implies that there is unique locally differentiable curve

$$x_i = x_i(x_1), \qquad i = 2, \ldots, n+1$$

defined in some neighborhood $(x_1 - \delta_{x_1}, x_1 + \delta_{x_1})$ of $x_1 \in \mathbf{x}_1$, and $(x_1 - \delta_{x_1}, x_1 + \delta_{x_1})$ is homeomorphic to

$$\pi^{-1}(x_1 - \delta_{x_1}, x_1 + \delta_{x_1}) = \{(x_1, x_2(x_1), \ldots, x_{n+1}(x_1)) | x_1 \in (x_1 - \delta_{x_1}, x_1 + \delta_{x_1})\}.$$

Therefore, since the point $(x_1, x_2(x_1), x_3(x_1), \ldots, x_{n+1}(x_1))^T$ is unique, π defines a one to one correspondence and thus a covering map from $\mathcal{Y}$ to $\mathbf{x}_1$. Since $\mathbf{x}_1$ is convex, $\alpha : I \to \mathbf{x}_1$, $\alpha(t) = (1-t)\text{inf}(\mathbf{x}_1) + t\text{sup}(\mathbf{x}_1)$ is a path in $\mathbf{x}_1$. Suppose

$$\pi(y_1) = \text{inf}(\mathbf{x}_1), \qquad \pi(y_2) = \text{sup}(\mathbf{x}_1).$$

By Proposition 9.1.2 in [2], there is a unique path

$$\tilde{\alpha} : [0,1] \to \mathcal{Y}$$

such that $\tilde{\alpha}(0) = y_1$, and $\pi(\tilde{\alpha}(t)) = \alpha(t)$ for all $t \in [0,1]$. Since π is one to one and

$$\pi(\tilde{\alpha}(1)) = \alpha(1) = \text{sup}(\mathbf{x}_1),$$

we have

$$\tilde{\alpha}(1) = y_2.$$

That is, $\tilde{\alpha}$ is the unique smooth path that connects y_1 and y_2. $\square$

THEOREM 3.2. *Let* $\mathbf{x}$ *be the box described in Algorithm 2.3. If Assumption 1 is true and if*

$$\boldsymbol{\eta}_i - 1 > M_i = \max_{X \in \mathbf{x}} \left| Det\left(\frac{\partial H}{\partial X_{-i}}\right)\right| \Big/ \left| Det\left(\frac{\partial H}{\partial X_{-1}}\right)\right|,$$

for all $i \neq 1$ *and* $X \in \mathbf{x}$*, then any curve entering the face* (1) *of the box* $\mathbf{x}$ *at the point* X_0 *will leave the box* $\mathbf{x}$ *through the face* (2)*.*

Proof. Without loss of the generality, assume $B_1 > 0$ so that

$$x_1 = \inf(\mathbf{x}_1) = X_{0,1}.$$

Since $\dfrac{\partial H}{\partial X_{-1}}$ is nonsingular at X_0, the implicit function theorem implies that there is a $\delta_{\mathbf{x}_1}$ such that in the interval $\mathbf{x}_1^* = [X_{0,1}, X_{0,1} + \delta_{\mathbf{x}_1}]$, there is a unique continuously differentiable path

$$\{(x_1, x_2(x_1), \ldots, x_{n+1}(x_1))^T \mid x_1 \in \mathbf{x}_1^*\}.$$

By the chain rule,

$$\frac{dx_i}{dx_1} = -Det\left(\frac{\partial H}{\partial X_{-i}}\right) \Big/ Det\left(\frac{\partial H}{\partial X_{-1}}\right),$$

and by Assumption 1, H has continuous derivatives. Therefore, since $\dfrac{\partial H}{\partial X_{-1}}$ is nonsingular in $\mathbf{x}$, there must exist an m_i such that

$$\left| Det\left(\frac{\partial H}{\partial X_{-i}}\right)\right| \leq m_i,$$

and

$$\left| \mathrm{Det}\left(\frac{\partial H}{\partial X_{-1}}\right)\right| \geq m > 0.$$

Thus,

$$\left|\frac{dx_i}{dx_1}\right| \leq \frac{m_i}{m} = M_i,$$

in $[X_{0,1}, X_{0,1} + \delta_{\mathbf{x}_1}] = \mathbf{x}_1^*$. By the mean value theorem,

$$x_i(x_1) = x_i\big(X_{0,1}\big) + \frac{dx_i}{dx_1}(\xi_i)\big(x_1 - X_{0,1}\big), \qquad x_i \in \mathbf{x}_1^*.$$

Thus,

$$X_{0,i} - M_i(x_1 - X_{0,i}) < x_i(x_1),$$

and

$$X_{0,i} + M_i(x_1 - X_{0,i}) > x_i(x_1),$$

Therefore, by our construction $x_i(x_1)$ is contained within the two straight lines, and hence $x_i(x_1) \in \mathbf{x}_i$. Since this argument holds for all $i \neq 1$, the locally unique curve is contained in the box $\mathbf{x}$. □

Theorem 3.1 asserts that any curve entering the box at the present point X_0 will exit the box on the face opposite the one on which X_0 lies. The following corollary and theorem assert that this curve will be unique within the box.

Corollary 3.1. *Suppose Assumption 1 is true. Also suppose that*

$$\boldsymbol{\eta}_i - 1 > M_i = \max_{X\in\mathbf{x}} \left|\mathrm{Det}\left(\frac{\partial H}{\partial X_{-i}}\right)\right| \Big/ \left|\mathrm{Det}\left(\frac{\partial H}{\partial X_{-1}}\right)\right| \tag{3}$$

is true in the constructed box $\mathbf{x}$*, for all* $i \neq 1$ *and all* $X \in \mathbf{x}$*. Suppose that there is a unique solution to* $H(X) = 0$ *on each face* (1) *and* (2)*. Then there is a unique curve passing through face* (1) *and* (2)*.*[3]

Proof. Assumption 1 and Theorem 3.2 imply that there is at least one continuous curve in the wedge-shaped volume bounded by the box $\mathbf{x}$ and the hyperplanes corresponding to the two straight lines defined in the proof of theorem 3.2. By assumption, there is a unique curve passing through the left face (1) of the box, and Theorem 3.2 implies that this curve must pass through the right face (2) of the box. However, by an assumption of this theorem, only one such curve passes through the hyperplane (2). The assertion of the theorem immediately follows. □

Remark 1. If we could verify (3) easily, Corollary 3.1 would make it more efficient than the following theorem to check uniqueness, since there is less overestimation when one of the coordinates is held fixed. Verification of (3) should be easier using automatic differentiation; this will be the subject of future work.

Theorem 3.3. *Under Assumption 1, there is a curve passing through* X_0*. If we further assume that* H *has continuous derivatives up to order 2, so that the derivatives* $\dfrac{d^2x_i}{dx_1^2}$ *are continuous*[4]*. Assume that*

$$\boldsymbol{\eta}_i > \frac{1}{2}\max_{X\in\mathbf{x}}\left|\frac{d^2x_i}{dx_1^2}\right| |B_1|\delta$$

for all $i \neq 1$*. Then there is a unique curve passing through the faces* (1) *and* (2) *of the box* $\mathbf{x}$ *described in Algorithm 2.3.*

Proof. By Taylor's theorem,

$$x_i(x_1) = x_i(X_{0,1}) + \frac{dx_i}{dx_1}(x_1)(x_1 - X_{0,1}) + \frac{1}{2}\frac{d^2x_i}{dx_1^2}(\xi_i)\big(x_1 - X_{0,1}\big)^2$$

for all $i \neq 1$ and $x_1 \in \mathbf{x}_1^* = [X_{0,1}, X_{0,1} + \delta]$. In Algorithm 2.3 $p_{i,1}$ and $p_{i,2}$ are defined by

$$p_{i,1} = X_{0,i} + B_i\delta - \boldsymbol{\eta}_i\delta,$$
$$p_{i,2} = X_{0,i} + B_i\delta + \boldsymbol{\eta}_i\delta;$$

But by assumption,

$$\boldsymbol{\eta}_i > \frac{1}{2}\max_{X\in\mathbf{x}}\left|\frac{d^2x_i}{dx_1^2}\right|(x_1 - X_{0,1})$$

for all $i \neq 1$, and $x_1 \in \mathbf{x}_1^*$. Thus,

$$p_{i,1} < x_i(x_1) < p_{i,2},$$

[3]although other curves may intersect $\mathbf{x}$.

[4]Actually, in this situation the chain rule implies that $\dfrac{d^2x_i}{dx_1^2}$ can be expressed in terms of H and its derivatives.

and $x_i(x_1)$ is contained in the box $\mathbf{x}_i$. Applying this argument to each coordinate completes the proof of the theorem. $\square$

The above two theorems motivate that heuristic in Algorithms 2.4 which dictates that we increase the $\boldsymbol{\eta}_i$ to obtain inclusion. However, since $\boldsymbol{\eta}_i$ too large may cause deleterious overestimation in the interval Jacobian matrix and hence preclude computation of the preconditioner, we decrease $\boldsymbol{\eta}_i$ whenever a preconditioner cannot be computed.

Remark 1. Corollary 3.1 does not guarantee that there is a unique curve within the box, but only that there is a unique curve passing through the two faces (1) and (2).

Remark 2. If we can verify (3) easily, Corollary 3.1 makes it more efficient to check uniqueness, since there is less overestimation when one of the coordinates is held fixed. Verification of (3) should be easier using automatic differentiation; this will be the subject of future work.

The following theorem clarifies when our interval step control will succeed.

THEOREM 3.4. *Make Assumption 1. Also assume*

$$\eta_{\min} > \max\{2|B_1|, 2M_2\},$$

where M_2 is determined in the proof. Finally, assume that $H(X)$ is Lipschitz at the point X_0, an approximation to the most recently computed point on the curve. Assume that X_0 has been computed to sufficient accuracy to ensure that $0 \in \mathbf{H}(\mathbf{x}_0)$, where $\mathbf{x}_0$ is defined in the proof below. Then, for small enough δ, Algorithm 2.4 must be successful; that is, the overall algorithm (Algorithm 2.1) in conjunction with Algorithm 2.4 will adjust $\boldsymbol{\eta}$ and δ so that the inclusion

$$\tilde{\mathbf{x}}_{-1} \subseteq \operatorname{int}(\mathbf{x}_{-1})$$

will eventually hold. In particular, the largest δ which will be acceptable depends only on the second derivative tensor of H and on the parameters $\eta_{\min}$ and $\eta_{\max}$.

Proof. Without loss of generality, suppose $B_1 > 0$. Since $\dfrac{\partial H}{\partial X_{-1}}$ is nonsingular in $\mathbf{x}$, by the implicit function theorem, there is a unique locally differentiable curve $x_i = x_i(x_1)$, $i = 2, \ldots, n+1$ defined in some neighborhood $(X_{0,1}, X_{0,1} + \delta)$ of $X_{0,1}$. From the construction of the box $\mathbf{x}$, $X_{0,1} \in \mathbf{x}_1$. From the construction of the box $\mathbf{x}_0$ in Algorithm 2.4,

$$\mathbf{x}_{0,i} = \begin{cases} \mathbf{x}_i & \text{if} \quad i = 1, \\ X_{0,i} & \text{otherwise.} \end{cases}$$

Thus, $X_0 \in \mathbf{x}_0$. By assumption, $0 \in \mathbf{H}(\mathbf{x}_0)$, so

$$0 \in Y_i \mathbf{H}(\mathbf{x}_0).$$

The construction of Algorithm 2.3 also implies

$$\mathbf{x}_{0,j} = X_{0,j} \subset \mathbf{x}_j,$$

whence

$$0 \in \mathbf{H}'(\mathbf{x})_j(\mathbf{x}_j - \mathbf{x}_{0,j}).$$

For $i \neq 1$, $\mathbf{x}_{0,i} = X_{0,i}$, and a Gauss–Seidel sweep in Algorithm 2.4 may be written as

$$\overline{\mathbf{x}}_i = X_{0,i} - \left[Y_i \mathbf{H}(\mathbf{x}_0) + \sum_{\substack{j=2 \\ j\neq i}}^{n+1} [Y_i \mathbf{H}'(\mathbf{x})]_j (\mathbf{x}_j - X_{0,j}) \right] \Big/ [Y_i \mathbf{H}'(\mathbf{x})]_i. \tag{4}$$

Hence, the numerator of (4) contains 0. Also, from the definition of optimal preconditioner in [4], $\inf([Y_i\mathbf{H}'(\mathbf{x})]_i) = 1$. Letting $\omega(\mathbf{x})$ denote the width of the interval $\mathbf{x}$, this and the fact that zero is in the numerator of (4) imply

$$\omega(\overline{\mathbf{x}}_i - X_{0,i}) = \omega \left[Y_i \mathbf{H}(\mathbf{x}_0) + \sum_{\substack{j=2 \\ j\neq i}}^{n+1} [Y_i \mathbf{H}'(\mathbf{x})]_j (\mathbf{x}_j - X_{0,j}) \right].$$

Let $C = [H'_{-1}(X_0)]^{-1}$ and let C_i be the i-th row of C. Then, since 0 is in the numerator in (4), Y_i minimizes the $\omega(\overline{\mathbf{x}}_i)$ (see [4]), and it follows that

$$\omega(\overline{\mathbf{x}}_i - X_{0,i}) \leq \omega \left[C_i \mathbf{H}(\mathbf{x}_0) + \sum_{\substack{j=2 \\ j\neq i}}^{n+1} [C_i \mathbf{H}'(\mathbf{x})]_j (\mathbf{x}_j - X_{0,j}) \right].$$

By the Mean Value theorem, it is easy to see that

$$\mathbf{H}'(\mathbf{x}) \subset H'(X_0) + M_1 (1)_{n\times(n+1)} (\mathbf{x} - X_0),$$

for some positive number M_1, where $(1)_{n\times(n+1)}$ stands for the $n \times (n+1)$ matrix with all components equal to 1. By the Lipschitz condition,

$$\mathbf{H}(\mathbf{x}_0) \subset M_1' (1)_{n\times 1} (\mathbf{x}_1 - X_{0,1}),$$

for some positive number M_1', where $(1)_{n\times 1}$ stands for the $n \times 1$ matrix with all components equal to 1. Then,

$$\begin{aligned}
\omega(\overline{\mathbf{x}}_i - X_{0,i}) &\leq \omega(C_i M_1'(1)_{n\times 1}(\mathbf{x}_1 - X_{0,1})) + \\
&\quad + \omega\Big(\sum_{\substack{j=2 \\ j\neq i}}^{n+1} \Big[C_i \big(H'(X_0) + M_1 (1)_{n\times(n+1)} (\mathbf{x} - X_0) \big) \Big]_j (\mathbf{x}_j - X_{0,j}) \Big) \\
&\leq \omega(C_i M_1'(1)_{n\times 1}(\mathbf{x}_1 - X_{0,1})) + \\
&\quad + \omega\Big(\sum_{\substack{j=2 \\ j\neq i}}^{n+1} \Big[C_i M_1 (1)_{n\times(n+1)} (\mathbf{x} - X_0) \Big]_j (\mathbf{x}_j - X_{0,j}) \Big) \\
&\leq M_2 \big(\delta + \max_{j\neq i,1} [\omega(\mathbf{x}_j - X_{0,j})]^2 \big),
\end{aligned}$$

for some M_2 which depends on M_1, M_1' and C. By assumption,

$$\boldsymbol{\eta}_i > 2|B_1| = \max_{1\leq i\leq n+1} 2|B_i|.$$

The construction of Algorithm 2.3 then implies $X_{0,i} \in \text{int}(p_{i,1}, p_{i,2})$ and

$$\mathbf{x}_i = (p_{i,1}, p_{i,2}),$$

so

$$2\eta_{\min}\delta \le \omega(\mathbf{x}_j - X_{0,j}) = 2\boldsymbol{\eta}_j\delta \le 2\eta_{\max}\delta.$$

Therefore,

$$\begin{aligned}\omega(\overline{\mathbf{x}}_i - X_{0,i}) &\le M_2(4\delta^2\eta_{\max}^2 + \delta)\\ &\le M_2\frac{(4\delta\eta_{\max}^2+1)}{2\eta_{\min}}(2\eta_{\min}\delta)\\ &\le M_2\frac{(4\delta\eta_{\max}^2+1)}{2\eta_{\min}}\omega(\mathbf{x}_i - X_{0,i}).\end{aligned}$$

Letting $p = \min\{|X_{0,i} - \inf(\mathbf{x}_i)|, |X_{0,i} - \sup(\mathbf{x}_i)|\}$ and observing that $X_{0,i} \in \text{int}(p_{i,1}, p_{i,2})$, it is easy to see from Algorithm 2.3 that

$$p = |\boldsymbol{\eta}_i - B_i|\delta.$$

The width of $\mathbf{x}_i$ is $2\boldsymbol{\eta}_i\delta$, so

$$\begin{aligned}\frac{\omega(\mathbf{x}_i - X_{0,i})}{p} &= \frac{2\boldsymbol{\eta}_i}{|\boldsymbol{\eta}_i - B_i|}\\ &\le 2 + \frac{2B_i}{\boldsymbol{\eta}_i - B_i}\\ &\le 2 + \frac{2B_i}{B_i}\\ &\le 4.\end{aligned}$$

If $M_2\big(\frac{4\delta\eta_{\max}^2+1}{2\eta_{\min}}\big) < \frac{1}{4}$, that is,

$$\delta < \Big(\frac{\eta_{\min}}{2M_2} - 1\Big)\frac{1}{4\eta_{\max}^2}, \tag{5}$$

then

$$\omega(\overline{\mathbf{x}}_i - X_{0,i}) \le p.$$

Since 0 is contained in the numerator of (4), $X_{0,i}$ is in both $\mathbf{x}_i$ and $\overline{\mathbf{x}}_i$, and M_2 is independent of δ. Thus, if δ is small enough to make (5) true, then

$$\overline{\mathbf{x}}_i \subseteq \text{int}(\mathbf{x}_i) \quad \text{for all } i \ne 1. \qquad \square$$

THEOREM 3.5. *Suppose i_0', $B_{i_0'}$, and $B1_{i_0'}$ are as in Algorithm 2.1. Then, after execution of Step 3 of Algorithm 2.1, B represents the same orientation along the curve as in the previous step.*

Proof. In the previous box, there is a unique curve that can be parametrized in terms of the previous parameter $x_{i_0'}$. However, if the i_0'-th component $B_{i_0'}$ changes sign within that box, then there must be a turning point with respect to i_0', which contradicts the fact that there is unique curve with respect to i_0' in the previous box. Therefore, to maintain orientation, it is necessary and sufficient that $\text{sgn}(B_{i'_0}) = \text{sgn}(B1_{i'_0})$, provided B and B_1 are both in the null space of H' at the previous and present points on the curve, respectively. □

Remark. If the null space of H' is computed using standard floating point arithmetic, then it cannot be rigorously verified that $\text{sgn}(B_{i'_0})$ is correct. However, this occurrence is highly unlikely at best, since $|B1_{i'_0}| = \max_{1\le i\le n+1} |B1_i|$. It is more likely that uniqueness could not be proven in this situation, and a smaller stepsize would then be taken. The algorithm can be made totally rigorous by computing the null space of H' using interval arithmetic.

4. Numerical Experiments.

As a rigorous step control method, interval step control guarantees that the curve followed is unique, i.e. that it is not possible to jump from one path to another. Thus, the interval step control follows the curve properly, regardless of how the initial, minimum, and maximum stepsizes are chosen. In contrast, success of approximate step controls depends on how we choose these parameters, and we may need to be overly conservative. The following numerical results illustrate this.

The experiments with the approximate step control were carried out with the software package PITCON 6.1. (See [8] and [9] for an explanation of the original version.) In this package, unless otherwise stated, we set the absolute error to 10^{-5} and relative error to 10^{-4}, and we did not request the algorithm to locate limit points. We supplied a subroutine with an analytic representation of the Jacobian matrix, rather than using finite differences. Furthermore, we allowed the program to choose the parameter coordinate. In the following tables, we use *steps* to indicate the number of predictor steps taken by the step controls from the starting point. Finally, we configured PITCON to reevaluate the Jacobian matrix after each step of the corrector iteration[5].

Comparisons in two dimensions. Many experiments revealed that, for two dimensional problems, the interval step control computed stepsizes that were somewhat large in relation to acceptable stepsizes in the approximate step control. For example, we tried the 2-dimensional Brown's almost linear curve and the 2-dimensional Layne Watson exponential cosine curve, defined by

(i) Brown's almost linear curve.

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

and

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

(ii) The Layne Watson exponential cosine curve.

$$f_i(X) = x_i - x_{n+1}e^{\cos\left\{i\left(\sum_{1\le j\le n} x_j\right)\right\}}, \quad 1 \le i \le n.$$

[5]That is, we configured PITCON to use the classical Newton's method as its corrector iteration.

The following table indicates the number of new points computed by the algorithms and the average stepsize used to follow the curves corresponding to the above two functions from $x_3 = 0$ to $x_3 = 1$. In PITCON 6.1 (the approximate algorithm), the minimum stepsize was set to 0.001, maximum stepsize to 0.02, and initial stepsize to 0.01.

Brown's almost linear function

	steps	average stepsize
interval	119	0.19674×10^{-1}
approximate	95	0.19773×10^{-1}

Layne Watson exponential cosine function

	steps	average stepsize
interval	110	0.11901×10^{-1}
approximate	84	0.19762×10^{-1}

Table 4.1. Results for $n = 2$.

Behavior as the dimension increases.

Consider the following initial-boundary problem:

$$\begin{cases} y'' + \lambda e^y = 0 \\ y(0) = 0, \quad y'(1) = 0. \end{cases}$$

Letting N be the number of the mesh points, one may discretize above problem into:

$$H(X) = 0$$

where

$$H_i(X) = \begin{cases} x_1, & i = 1, \\ x_{i-1} - 2x_i + x_{i+1} + x_N e^{x_i} * d, & i = 2, \dots N-2, \\ x_{N-1} - x_{N-2}, & i = N-1, \end{cases}$$

where $d = 1/(N-2)^2$ and $x_N = \lambda$. To include portions of the curve with changing curvature, we follow the curve from $\lambda = 0$ and $x_i = 0$, $i = 1, \dots, N-1$, past a turning point of x_N with respect to x_{N-1}. This turning point occurs at $x_N \approx .9$. See figure 4.3, where $N = 60$.

The following table shows how the interval step control behaves as the number of mesh points increases. In the table, CPU ratio is used to indicate the ratio of CPU times of experiments corresponding to the previous and present rows of the table. We set the maximum stepsize in both PITCON and the interval step control to 0.2×10^{-1}. This is a reasonable stepsize for obtaining an accurate plot.

N	steps	average δ	CPU(s)	CPU ratio	N^3 ratio
10	252	0.19910×10^{-1}	27.35		
20	315	0.19921×10^{-1}	198.00	7.24	8
30	371	0.19910×10^{-1}	732.19	3.70	3.37
40	420	0.19950×10^{-1}	1968.17	2.69	2.37
50	464	0.19954×10^{-1}	4425.24	2.25	1.95
60	504	0.19957×10^{-1}	8510.62	1.92	1.73

Table 4.2a. Results for discretized mesh problem with interval step control.

From the above Table, we see that the CPU time increases approximately as the cube of N. This is because, in our implementation, we are using a dense LP solver, and we are handling the interval Jacobi matrix as a dense matrix. Using a banded solver and a band (or sparse) structure in the interval Jacobi matrix should lead to an increase of only order N^2.

The following table was obtained from PITCON applied to the same problem, with the same maximum stepsize. We report results both with full storage user Jacobian matrix and band storage central difference Jacobian matrix. Since PITCON algorithms with either full storage user Jacobian or band storage central difference Jacobian give the same number of steps and average step size, only different CPUs are given in the following table.

N	steps	average δ	CPU(s)-full	CPU(s)-band
10	251	0.19960×10^{-1}	0.37	0.49
20	317	0.19968×10^{-1}	0.98	1.05
30	373	0.19973×10^{-1}	2.09	1.74
40	423	0.19976×10^{-1}	4.74	2.56
50	467	0.19976×10^{-1}	6.03	3.46
60	507	0.19980×10^{-1}	9.04	4.48

Table 4.2b. Results for the discretized mesh problem with approximate step control: PITCON with full storage user Jacobian and band storage central difference Jacobian.

The above table shows low CPU time and weak dependence on N for PITCON. However, the interval arithmetic in Fortran-SC is implemented with subroutine calls. The CPU times should be substantially faster for hardware interval arithmetic. The following table indicates that on IBM3090 machine with ACRITH interval arithemetic is a factor of at least 15 slower than real arithemetic.

experiment	CPU ratio of interval and real
$\sum_1^{10^6} 1.0$	39.7
$\prod_1^{10^6} 1.0$	34.6
compute sin(1.0) 10^6 times	17.7

Table 4.2c. CPU ratios for simple computations with ACRITH and with double precision arithmetic.

Interval arithematic in micro code can be only a factor of 2 slower than floating point arithematic, and good compiler may provide code for which the factor is only 5 or less.

Behavior on the discretized problem with bigger maximum stepsize, for $N = 60$.

We reran the programs for the discretized problem with $N = 60$, trying both PITCON and the interval step control with the same initial configuration, except with a maximum stepsize of 0.5 instead of 0.02, as it was in Tables 4.2a and 4.2b. We set the initial stepsize to 0.05. The following is table indicates the behaviors of both step controls.

step controls	steps	average δ	CPU(s)
interval step control	23	0.44401	487.736
approximate step control	26	0.45357	0.5541

Table 4.2d. Results for the discretized mesh problem, $N = 60$, with interval step control and PITCON with full storage user Jacobian and band storage central difference Jacobian.

Behavior of discretized problem near a solution tending to infinity as λ tends to 0..

After the curve passes the local maximum, λ tends to zero, while the other components of the solution X tend to infinity, and the condition number of the Jacobian matrix tends to infinity. These conditions cause the stepsize in interval step control to become smaller. However, PITCON follows the curve without decreasing the stepsize. However, the experiment shows that PITCON also follows the curve. Tables 4.2d and 4.2e show how the stepsize δ is decreased as a function of λ, for each of these methods.

steps	λ	stepsize δ
10×10^1	8.6834×10^{-1}	2.0×10^{-2}
10×10^2	8.0191×10^{-2}	2.0×10^{-2}
12×10^2	1.6420×10^{-3}	2.0×10^{-2}
15×10^2	1.3567×10^{-4}	2.0×10^{-2}
18×10^2	1.0214×10^{-5}	2.0×10^{-2}
20×10^2	1.7596×10^{-6}	2.0×10^{-2}

Table 4.2e. Results from PITCON, for the discretized mesh problem near a singularity, with full storage user Jacobian and $N = 10$.

steps	λ	stepsize δ
10×10^1	8.9672×10^{-1}	2.0×10^{-2}
10×10^2	2.3591×10^{-2}	1.3×10^{-3}
20×10^2	9.4530×10^{-3}	8.7×10^{-4}
15×10^3	8.7438×10^{-4}	9.6×10^{-5}
20×10^3	6.2045×10^{-5}	7.3×10^{-5}

Table 4.2f. Results for the discretized mesh problem near a singularity with interval step control and $N = 10$.

When λ tends to zero, the Jacobian matrix is singular enough to let interval step control to reduce the stepsize, but the approximate step control PITCON does not reduce the stepsize.

More dimension dependence: Brown's almost linear function..

In a separate experiment to observe the dependence of the interval step control on the dimension of the problem, we ran experiments with Brown's almost linear function for $n = 5$, $n = 10$, $n = 15$ and $n = 20$. The following table gives average stepsizes and numbers of new points computed for the 5-, 10-, 15-, and 20- dimensional Brown's curve, using the interval step control. The maximum allowable stepsize never became binding during the algorithm's execution. The table gives the average stepsize which the algorithm actually used to get from $x_{n+1} = 0$ to $x_{n+1} = 1$. These experiments illustrate that, for some problems, the computational work required to use the interval step control does not increase unduly with dimension. (However, the work associated with the linear algebra in our present implementation increases relatively rapidly, especially for the banded problems.)

n	steps	average δ
5	403	0.3747×10^{-2}
10	163	0.1015×10^{-1}
15	332	0.5214×10^{-2}
20	498	0.3575×10^{-2}

Table 4.3. Results for Brown's function.

Behavior of the Layne Watson exponential function $n = 5$.

We ran interval step control for the Layne Watson exponential cosine curve for $n = 5$. The algorithm took $39,896$ steps with an average stepsize of 0.2843×10^{-3}. Though, in this case, the stepsize is somewhat smaller, it is still reasonable when the interval step control is used.

Since, in practical problems, it is hard to know beforehand how rapidly the curvature will change, it not easy to decide which maximum allowable stepsize is appropriate for a specific problem. For example, when we use large initial or large maximum allowable stepsizes in PITCON 6.1, the resulting path following is not successful, though these stepsizes pose no problem for the interval step control.

In all cases, we used maximum stepsize in PITCON equal to 2.5. This is the value to which it was set in the test drivers packaged with PITCON. This maximum stepsize is large in relation to the portion of arc we wish to follow. Thus, the algorithm's behavior will depend on the step control, and not on the maximum stepsize.

Figure 4.1 gives graphs obtained for the Layne Watson exponential function with $n = 5$, from both Algorithm 2.1 and PITCON. In PITCON, the initial stepsize was set equal to 0.1, while the minimum stepsize was set equal to 10^{-7}. Our interval step control had no minimum stepsize, *and the minimum stepsize never became binding in PITCON. In other words, the actual stepsizes that PITCON took would not change, even if we made the minimum allowable stepsize smaller. Rerunning the problem using PITCON with minimum stepsizes up to* .01*, we obtained exactly the same sequence of steps.*

We reran PITCON with minimum stepsize equal to 10^{-7} and initial stepsize equal to 10^{-4}. Surprisingly, a worse problem than that in figure 4.1 occurred. PITCON lost orientation near the beginning of the curve, and the computed points had coordinates x_5 and x_6 which were negative. However, when we reran PITCON with minimum stepsize equal to 10^{-7} and initial stepsize equal to 10^{-7}, the algorithm behaved nicely. In the latter case, it produced a plot similar to the interval step control, and took only 48 steps.

With a minimum stepsize of 10^{-10} and an initial stepsize of 10^{-8}, PITCON completed in 20 steps. However, the actual curve has four local maxima of x_5 with respect to x_6, and PITCON skipped the one closest to the starting point. (See figure 4.1b.)

The point we wish to make is not that approximate step controls cannot do well, but only that they may require careful analysis of the output and extensive interaction to be confident of the results. A faithful rendering of the graph of the curve would occur with an approximate step control with a sufficiently small maximum stepsize. However, how small is small enough may not be clear beforehand.

Behavior on the topologist's sine curve.

The topologist's sine curve is defined as

$$\{(x, t) = (x, \sin(1/x)) \mid x \neq 0\}.$$

It is well known that the curve has a severe oscillation near $x = 0$, and approximate curve following algorithms tend to "skip" or lose orientation. The graphs in Figure 4.3 were obtained by running Algorithm 2.1 and PITCON with starting point $(0.019, \sin(1/0.019))$. In PITCON, we used initial stepsize 0.1, minimum stepsize 0.05 and maximum stepsize 0.5.

This is an artificial problem. However, changes in curvature become more severe as x tends to 0, so it is a reasonable test of curve-tracking. The interval step control can adjust to these changes in curvature, increasing and decreasing the stepsize as appropriate. If we wish to follow the curve to x a specified distance from 0, we can set the maximum stepsize in PITCON small enough to make it as successful as interval step control. We can estimate that, at $x = 0.006$, where we let interval continuation stop, the maximum stepsize can be no larger than 10^{-4}. However, using this as the maximum stepsize from the starting point would cause PITCON to proceed very slowly.

We can also take another point of view: The interval algorithm with the same initial stepsize and maximum stepsize as we used in PITCON proceeds successfully past the point where PITCON 6.1 started to "skip." This is because the interval step control guarantees that there is only one point on the curve in the slice of the box perpendicular to the parameter coordinate. Therefore, when the algorithm approaches a point where the curvature is larger, it decreases the stepsize as much as necessary, then increases the stepsize after passing this point. But approximate step controls do not always correctly anticipate such points.

Behavior on a parametrized family of hyperbolas.

Define a parametrized family of functions $f(x,t) : \mathbb{R}^2 \to \mathbb{R}$ by

$$f(x,t) = x^2 - (t - 0.5)^2 - p^2,$$

where p is a shape parameter. $f = 0$ defines the simple hyperbolic curve. The curve has two disjoint branches, with vertices $(-p, 0.5)$ and $(p, 0.5)$. As p tends to 0, the two branches become closer and closer, and the curve degenerates into two straight lines, as in the left graph of Figure 4.4.

This experiment was done by setting $p = 10^{-5}$ and using starting point $(0.5, 0)$, initial stepsize 0.01, minimum stepsize 0.0001, and maximum stepsize 0.01. In addition we tried $p = 10^{-15}$; in this case, the distance between the two vertices is only $2p = 2 \times 10^{-15}$. But the interval step control is still successful. We note that 10^{-16} is near the double precision machine epsilon on the IBM3090.

The family of hyperbolas can be viewed as an imperfect bifurcation, in which the parameter controls both the change in curvature and the distance from a true bifurcation problem. In situations where it is reasonable to interpret such an imperfect bifurcation as a bifurcation, the behavior of the approximate step control may be acceptable, provided not all branches are required. In all cases, the interval step control correctly renders the actual mathematical curve, or else stops short, indicating that it cannot proceed further.

The following table gives x, t and the stepsize δ near the vertex $(p, 0.5)$, for $p = 10^{-15}$, for our interval step control. This table does not include all points (x, t) which were computed within this range, but is meant to show the way that the stepsize is decreased harmonically as t tends to 0.5.

x	t	δ—stepsize
0.48612×10^{-00}	$0.01318750000000000 \times 10^{00}$	0.75937×10^{-02}
0.36692×10^{-04}	$0.49996406427866499 \times 10^{00}$	0.75608×10^{-06}
0.29184×10^{-09}	$\underline{0.499999999}70816975 \times 10^{00}$	0.25637×10^{-11}
0.31129×10^{-10}	$\underline{0.4999999999}6887090 \times 10^{00}$	0.28518×10^{-12}
0.27481×10^{-11}	$\underline{0.49999999999}725195 \times 10^{00}$	0.56908×10^{-13}
0.47811×10^{-12}	$\underline{0.499999999999}52188 \times 10^{00}$	0.29706×10^{-14}
0.67378×10^{-13}	$\underline{0.4999999999999}3297 \times 10^{00}$	0.27818×10^{-15}
0.61542×10^{-14}	$\underline{0.49999999999999}398 \times 10^{00}$	0.73431×10^{-16}
0.10834×10^{-14}	$\underline{0.500000000000000}40 \times 10^{00}$	0.29203×10^{-16}
0.57733×10^{-14}	$\underline{0.50000000000000}562 \times 10^{00}$	0.40494×10^{-15}
0.24800×10^{-13}	$\underline{0.5000000000000}2445 \times 10^{00}$	0.18997×10^{-14}
0.32370×10^{-12}	$\underline{0.500000000000}31981 \times 10^{00}$	0.23674×10^{-13}
0.20241×10^{-11}	$\underline{0.50000000000}200362 \times 10^{00}$	0.16346×10^{-12}
0.39149×10^{-10}	$\underline{0.5000000000}3863578 \times 10^{00}$	0.29980×10^{-11}
0.14356×10^{-04}	$0.50001435591026541 \times 10^{00}$	0.10581×10^{-06}
0.27344×10^{-01}	$0.52734443705535122 \times 10^{00}$	0.20037×10^{-02}

Table 4.4. Around a hyperbola's vertex.

Figure 4.1. LAYNE WATSON EXPONENTIAL FUNCTION WITH $n = 5$.

5. Summary, Conclusions and Future Work.
We have presented algorithms for interval step control for continuation methods. We have developed a theory to clarify how these algorithms are qualitatively more reliable than algorithms based on approximate step controls. We have presented several

Figure 4.1B. PITCON WITH INITIAL STEPSIZE 10^{-8}, MINIMUM STEPSIZE 10^{-10}, AND $n = 5$.

Figure 4.2. DISCRETIZED MESH PROBLEM$(N = 60)$ WITH INTERVAL STEP CONTROL.

classes of numerical experiments that illustrate the immunity of these interval methods to certain types of failures, that illustrate that these methods can also be practical on various problems, and that show that dimension is not an intrinsic limitation.

Figure 4.3. Topologist's sine curve.

Figure 4.4. Almost degenerate hyperbola.

State-of-the-art implementations of continuation methods based on non-interval step controls function efficiently and fairly predictably, provided sufficient knowledge is known about the problem to set the algorithm tolerances appropriately, or provided it is possible to repeatedly redo the computation after human interaction. However, interval step controls should be considered in cases in which rigor is critical or in which human interaction with the method is not possible.

If $n = 1$, a separate idea that substantially reduces the inherent overestimation may be used. We are developing this method and associated software. Though less universal than the general case, this software will be both rigorous and efficient.

Further improvements in efficiency are undoubtedly possible. Also, methods to estimate the bounds M_i and second derivative bounds appearing in Theorems 3.2 through 3.5 will allow for an algorithm that, besides not giving erroneous results, will always succeed in following the curve, without human intervention.

The interval step control in this paper makes use of an explicit representation of the Jacobi matrix for the system. Also, rows of a matrix similar to the inverse of the Jacobi matrix are computed and used, one row at a time. Though the entire inverse does not need to be stored (only one row at a time does), the amount of computation may increase more rapidly in n than with some approximate step controls. For example, quasi-Newton updates can be used with approximate step controls, but do not make sense with an interval step control. In principle, automatic differentiation may be used to obtain numerical values of an "analytic" Jacobi matrix, in fairly general contexts, such as when the function is given by an involved subroutine. However, it is unclear when this will be practical. In practice, there are many continuation problems with explicit representation by analytic expressions. These include, for example, finding the roots of a polynomial system with moderate n, or finding the solutions of some of the systems obtained by discretization of boundary value problems.

REFERENCES

[1] Allgower, E., and Georg, K., *Numerical Continuation Methods: An Introduction*, Springer-Verlag, New York, 1990.

[2] Brown, R., *Topology*, Horwood, Chichester, England, 1988.

[3] Heijer, C. den, and Rheinboldt, W. C., *On Steplength Algorithms for a Class of Continuation Methods*, SIAM J. Numer. Anal. 18 no. 5 (1981), pp. 925–948.

[4] Kearfott, R. B., *Preconditioners for the Interval Gauss–Seidel Method*, SIAM J. Numer. Anal. 27 no. 3 (1990), pp. 804–822.

[5] Kearfott, R. B., Hu, C. Y., Novoa, M. III, *A Review of Preconditioners for the Interval Gauss–Seidel Method*, Interval Computations 1 no. 1 (1991), pp. 59–85.

[6] Neumaier, A., *The Enclosure of Solutions of Parameter-Dependent Systems of Equations*, Reliability in Computing: The Role of Interval Methods in Scientific Computing, Academic Press, New York, etc., 1988, pp. 269–86.

[7] Neumaier, A., *Interval Methods for Systems of Equations*, Cambridge University Press, Cambridge, England, 1990.

[8] Rheinboldt, W. C., Burkardt, J. V., *A Locally Parameterized Continuation Process*, ACM Trans. Math. Software 9 no. 2 (1983), pp. 215–235.

[9] Rheinboldt, W. C., and Burkardt, J. V., *A Program for a Locally-Parametrized Continuation Process*, ACM Trans. Math. Software 9 no. 2 (1983), pp. 215–35.