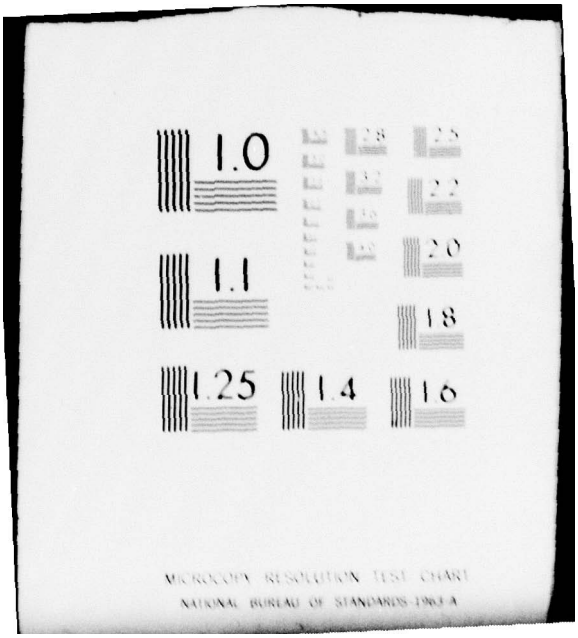
AD-A077 129 WISCONSIN UNIV-MADISON MATHEMATICS RESEARCH CENTER F/G 12/1
A SIMPLE RELIABLE NUMERICAL ALGORITHM FOR FOLLOWING HOMOTOPY PA--ETC(U)
JUL 79 T LI , J A YORKE DAAG29-75-C-0024
UNCLASSIFIED MRC-TSR-1984 NL

| OF |
AD
A077129

END
DATE
FILMED
12-79
DDC


1.0
1.1
1.25
1.4
1.6
2.0
1.8
2.2
2.5
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS 1963-A

MRC Technical Summary Report #1984

A SIMPLE RELIABLE NUMERICAL ALGORITHM FOR FOLLOWING HOMOTOPY PATHS

See 1473 in back

Tien-Yien Li and James A. Yorke

Mathematics Research Center
University of Wisconsin—Madison
610 Walnut Street
Madison, Wisconsin 53706

July 1979

(Received June 20, 1979)

Approved for public release
Distribution unlimited

Sponsored by

U. S. Army Research Office
P. O. Box 12211
Research Triangle Park
North Carolina 27709

and

National Science Foundation
Washington, D. C. 20550

UNIVERSITY OF WISCONSIN - MADISON
MATHEMATICS RESEARCH CENTER

A SIMPLE RELIABLE NUMERICAL ALGORITHM FOR FOLLOWING HOMOTOPY PATHS

Tien-Yien Li[1],[3] and James A. Yorke[2],[4]

Technical Summary Report # 1984
July 1979

ABSTRACT

The purpose of this paper is to develop simple algorithms and flow charts for the homotopy continuation methods so that scientists may easily program reliable curve followers. The emphasis is on simplicity and reliability at the expense of the speed.

AMS (MOS) Subject Classiciations: 90C99, 65H10

Key Words: Homotopy, path-following, continuation methods.

Work Unit Number 2 - Other Mathematical Methods
and
Work Unit Number 5 - Mathematical Programming and Operations Research

(1) On leave from the Department of Mathematics, Michigan State University, East Lansing, Michigan 28824.

(2) Institute for Physical Science and Technology, University of Maryland, College Park, Maryland 20742.

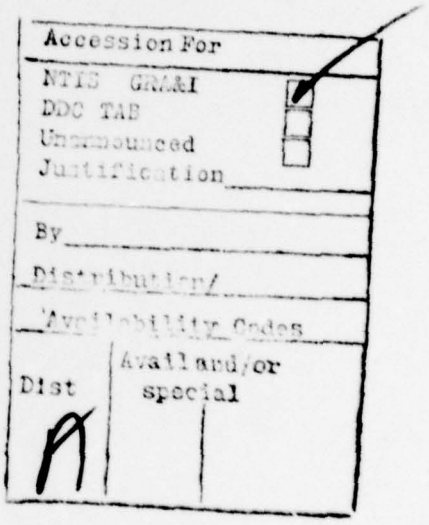
Sponsored by the United States Army under Contract No. DAAG29-75-C-0024. This material is based upon work supported by:

3) National Science Foundation Grant Nos. MCS78-02420 and MCS78-09525;

4) National Science Foundation Grant No. MCS76-24432.

SIGNIFICANCE AND EXPLANATION

The essence of the "continuation method" is "path following". The purpose of this paper is to develop simple algorithms and flow charts so that scientists may easily program reliable curve followers.



A SIMPLE RELIABLE NUMERICAL ALGORITHM FOR FOLLOWING HOMOTOPY PATHS

Tien-Yien Li[1],[3] and James A. Yorke[2],[4]

§1. Introduction

Newton's method has probably long been the most frequently used method for solving systems of nonlinear equations but it is most useful when an approximate solution is available. With the recent work of Scarf [11], Eaves [7], Saigal [6] and others, powerful new techniques have become available for the solution of systems of nonlinear equations. Their approach is based on simplicial decompositions of spaces involved and the following of a piecewise linear path to the solution. Kellogg, Li and Yorke [10], followed by Smale [12], and Chow, Mallet-Paret, and Yorke [3] found a related approach for systems of smooth (differentiable) systems in which the piecewise linear path was replaced by a smooth curve. In this approach simplicial decomposition and pivoting methods are replaced by curve following techniques which are based on highly developed nonlinear analysis techniques including Newton's method and ordinary differential equations solvers, which seem to be easier to implement. These methods are generally known as "continuation method".

The essence of the "continuation method" is "path following". The purpose of this paper is to develop simple algorithms and flow charts so that scientists may easily program reliable curve followers. The emphasis is on <u>simplicity</u> and

(1) On leave from the Department of Mathematics, Michigan State University, East Lansing, Michigan 28824.

(2) Institute for Physical Sciences and Technology, University of Maryland, College Park, Maryland 20742.

Sponsored by the United States Army under Contract No. DAAG29-75-C-0024. This material is based upon work supported by:

3) National Science Foundation Grant Nos. MCS78-02420 and MCS78-09525.

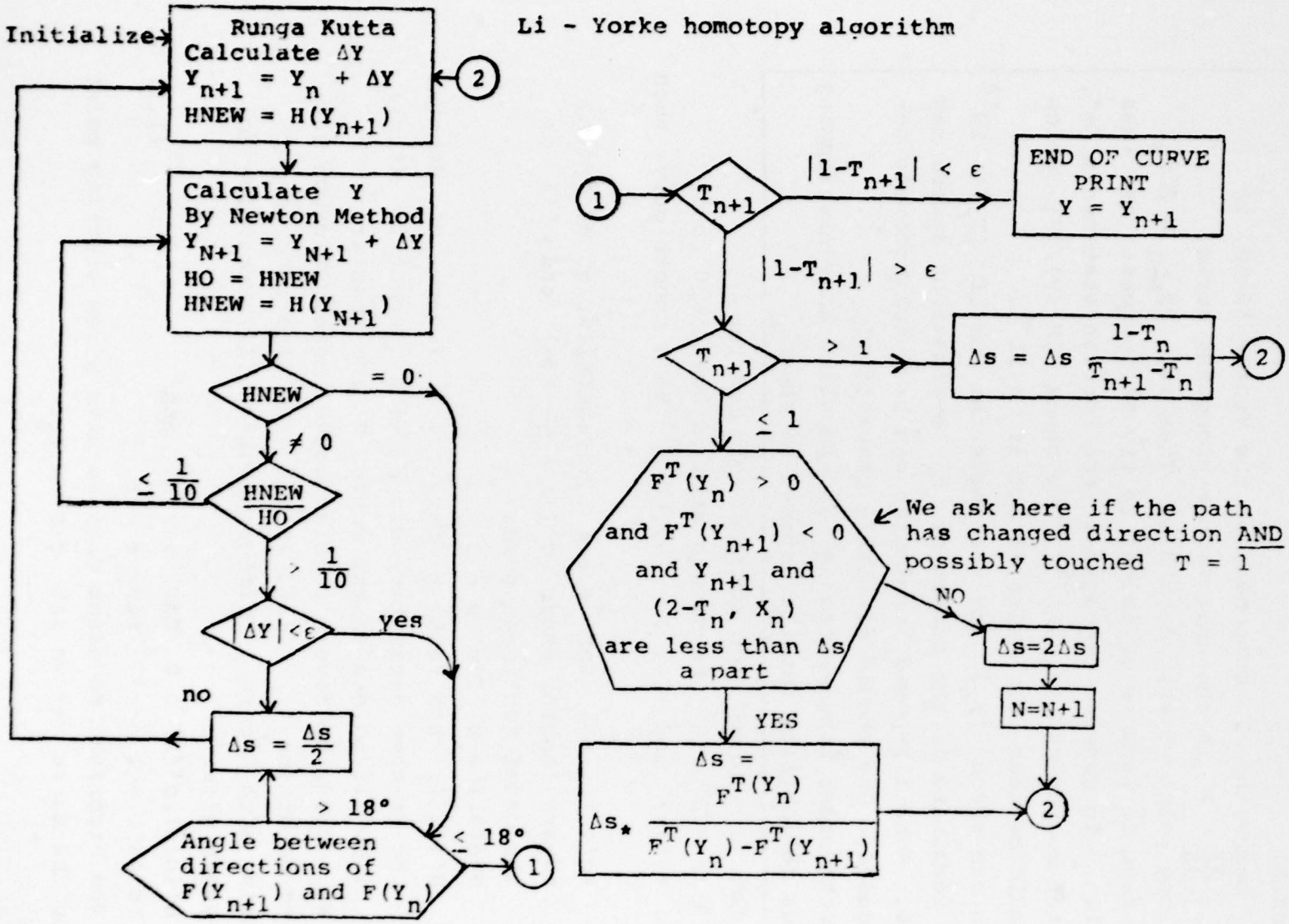
4) National Science Foundation Grant No. MCS76-24432.

reliability at the expense of speed. We find that simple routines using variable step-size differential equation solvers can be made reliable. Our preliminary experiments have utilized a fourth order Runge-Kutta solver in which we have found it necessary in some cases to vary rapidly the step length over a range of 10^{-10}.

The philosophy of the approaches of Watson [13], Smale [12] and others may be interpreted as minimizing the importance of accuracy in following the curve. In lectures Smale has advocated large jumps with little concern for whether the follower accidentally skips from one path of zeros to another nearby one. Now, however, classes of problems are coming to light in which it is important to remain on a simple curve. As an example, consider the problem of solving all the roots of a system of n polynomials in n complex variables [4] [5] , [8] , [9] , by homotopy method. It is important to note that it is necessary to follow a different curve to obtain each solution. If the paths are not followed with care, solutions can be missed. The same necessity occurs with other problems such as the inverse eigenvalue problem [1] . The curve must be followed carefully.

Paragraphs 2 - 4 describe an algorithm which is summarized in the following flow chart for following a nondegenerate curve to a solution. The existence of such curves is "guaranteed with probability one" in [3] . Nonetheless §5 describes situations in which the algorithm may fail to follow the curve to a point desired. In §6 numerical results are described.

For convenience, we limit our discussion here to the homotopy type algorithms as required by Chow, Mallet-Paret and Yorke [3] , Alexander and Yorke [2] . To follow the solution curves in the algorithms based on the global Newton method of Smale [12] or the non-retraction method given by Kellogg, Li and Yorke [10] are quite similar with some variations. In all these cases we follow a solution curve by numerically integrating an initial value problem until a point of interest is encountered.


Li - Yorke homotopy algorithm
Initialize
Runga Kutta
Calculate ΔY
$Y_{n+1} = Y_n + \Delta Y$
$HNEW = H(Y_{n+1})$
2
Calculate Y
By Newton Method
$Y_{N+1} = Y_{N+1} + \Delta Y$
$HO = HNEW$
$HNEW = H(Y_{N+1})$
HNEW
= 0
≠ 0
$\leq \frac{1}{10}$
$\frac{HNEW}{HO}$
$> \frac{1}{10}$
$|\Delta Y| < \varepsilon$
yes
no
$\Delta s = \frac{\Delta s}{2}$
> 18°
Angle between directions of $F(Y_{n+1})$ and $F(Y_n)$
≤ 18°
1
T_{n+1}
$|1-T_{n+1}| < \varepsilon$
END OF CURVE
PRINT
$Y = Y_{n+1}$
$|1-T_{n+1}| > \varepsilon$
T_{n+1}
> 1
$\Delta s = \Delta s \frac{1-T_n}{T_{n+1}-T_n}$
2
≤ 1
$F^T(Y_n) > 0$
and $F^T(Y_{n+1}) < 0$
and Y_{n+1} and
$(2-T_n, X_n)$
are less than Δs
a part
We ask here if the path has changed direction AND possibly touched T = 1
NO
Δs=2Δs
N=N+1
YES
$\Delta s = \Delta s_* \frac{F^T(Y_n)}{F^T(Y_n)-F^T(Y_{n+1})}$
2

COMMENTS AND REFINEMENTS.

1. Write $Y_n = (T_n, X_n)$
2. Initialize $T_0 = 0$
   X_0 = initial point
   $n = 0$, $\Delta s = .1$, $\varepsilon = 10^{-14}$ for an 18 digit computer word length.
3. F^T means the T component of the vector field, if $F^T = \frac{dT}{ds}$. F is the unit vector along the curve.
4. At one point we set $\Delta s = 2\Delta s$. That is, $\Delta s_{M+1} = 2\Delta s_M$. In fact it is more efficient to try to increase Δs less often. In general "equations" are Fortran statements "=".
5. After computing $\Delta Y = H'^{-1}H$, we check $|H'\Delta y|/|H|$ which should be near 0. Print it out if $> 10^{-4}$.
6. One can check $|Y_n|$ and terminate the run if $|Y_n| > 10^{10}$, or terminate if the number of H' evaluations exceed say 500. Some ill-posed homotopies can have trajectories unbounded. One should similarly check if $T_{n+1} < 0$.
7. Fourth order Runga-Kutta is used to allow easy programming plus reliability (at the expense of speed).

Let $J = [0,a]$ for some $a > 0$ and let $H : [0,1] \times R^n \to R^n$ be smooth with $H(0,x_0) = 0$ and $H(1,x) = F(x)$ and $C : J \to [0,1] \times R^n$ be a smooth curve such that

(i) $\|c'(s)\|_2 = 1$ for $s \in R$ (parametrization according to arc length) where $c'(\cdot) = \frac{d}{ds} c(\cdot)$ and $\|\cdot\|$ is the usual Euclidean norm.

(ii) $H(c(s)) = 0$ for $s \in J$

(iii) $H'(c(s))$ has rank n for $s \in J$ (where H' denotes the Frechet derivative of H with respect to (t,x)) i.e. all points on the curve C are regular.

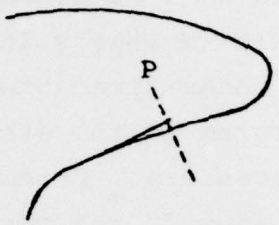
We suppose we have a regular curve, namely a curve satisfying condition (i), (ii), (iii). The algorithm is for following such a curve. Differentiating (ii) according to arc length gives

$$H'(c(s))c'(s) = 0 \quad \text{for } s \in R, \text{ and} \qquad \|c'(s)\| = 1 \quad \text{for } s \in R \ . \tag{1.1}$$

The algorithm requires that we have given starting point y^0 on the curve and we set $c(0) = y_0$.

§2. Proceeding along the curve

When integrating the differential equation (1.1) numerically, one inevitably makes some error which will become significant if one does not wish to use a small step size. By taking into account the special form of our problem, it is possible to correct the integration procedure. This is done by choosing a hyperplane P approximately "perpendicular" to the curve and calculating the locally unique point in $H^{-1}(0) \cap P$ by Newton method as illustrated in the Figure.



Thus one can consider the integration procedure as a "predictor step" and the Newton method as a "corrector step". Theoretically, there exists an S_0 such that if the "step size" jump s is less than S_0 then the Newton method will always converge [14]. In practice, we take an initial jump of length s_1 from y_m reaching at a point z^1, which may or may not be in the convergence region of Newton method. In our algorithm, we first make a Newton step starting at z^1, reach a point z^2. We then compare $\|H(z^1)\|$ and $\|H(z^2)\|$. If $\|H(z^1)\| > 10\|H(z^2)\|$, we then use Newton method repeatedly to generate $z^3, z^4, \cdots$. Before using Newton method at z^n we check to see if

$$\|H(z^{n-1})\| > 10\,\|H(z^n)\| \tag{2.1}$$

is satisfied. If (2.1) is not satisfied we will check to see if $\|\Delta z_n\| = \|z^n - z^{n-1}\|$ is too small to make a significant improvement in H values because the rounding off of the machine.

Therefore, let ε be the machine accuracy. If $\|\Delta z_n\| < \varepsilon$ we set $y_{m+1} = z_n$ and move on. If $\|\Delta z_n\| > \varepsilon$, the initial step size s_1 will be cut in half and the whole procedure is restarted with the new step size. To save the number of the Newton iterations, the 4^{th}-order Runge-Kutta

method is used as a "predictor". (It is 5^{th}-order on each step.) Therefore, every time the step size is cut in half the "predicted" point is expected to be roughly 32 times "closer" to the solution curve than it was before. The number of steps required could clearly be reduced slightly by using rough rules of thumb on how many times to use Newton's method before taking the next step, but in practice the difference will be small and such rules are difficult to justify in a general purpose algorithm. On the other hand if a lower order solver was used as a predictor, the required number of steps would increase immensely in some of the problems we have investigated. Adams predictors are substantially more difficult to program if the step size is to be varied rapidly as we find necessary. Standard Adams differential equations solver packages do not permit one to intersperse Newton method correctors after each step, and because of slow start-up routines these packages are only efficient when the Newton correctors are used after many steps, but then it is necessary to take small steps so that the cummulative error remains small.

The solution would be to make major surgical changes on a standard package and create a new package. The usual difficulty with Runge-Kutta Schemes is that it is difficult to choose optimal step size is eliminated in our situation by the existence alternative estimates on the optimality described in this section and the next.

§3. Angle checking

From (1.1), $c'(s)$ is characterized by $H'(c(s))c'(s) = 0$ and $\|c'(s)\| = 1$ and c' changes continuously. Since H' is an $n \times (n+1)$ matrix with rank n, there exists two vectors, differing by a factor of -1, v and $-v$, such that $H'v = -H'v = 0$ and $\|v\| = 1$. So, $c'(s) = v$ or $c'(s) = -v$. Choosing the wrong sign, may lead to "misorientation" as in the Figure.

By using the index theory, this error can be corrected by checking the sign of the determinants of a certain matrix. This method is inefficient, for the computation of determinants is unstable, expensive and generally unnecessary. Therefore, to move along the curve in a consistent way, we adopt the following method.

Let y_1, y_2 be two successive points on $c(s)$ with $H(c(s)) = 0$, and v^1, v^2 be the corresponding 0-vector of $H'(y_1)$, $H'(y_2)$ with $\|v^1\| = \|v^2\| = 1$. We choose the sign of v^2 to enforce the condition

$$\|v^1 - v^2\| \leq .3 \quad \text{or simply} \qquad (3.1)$$

$$\langle v^1, v^2 \rangle \geq .95$$

where $\langle , \rangle$ represents the usual inner product in the Eucledian norm. That is, we require the angle between v^1 and v^2 be less than about 18°. We cut the step size in half, if necessary, to fulfill the condition (3.1). By using this angle checking routine, our algorithm gives more points automatically on the curve when the curve has a "sharp" turn.

§4. Speed-up the step size

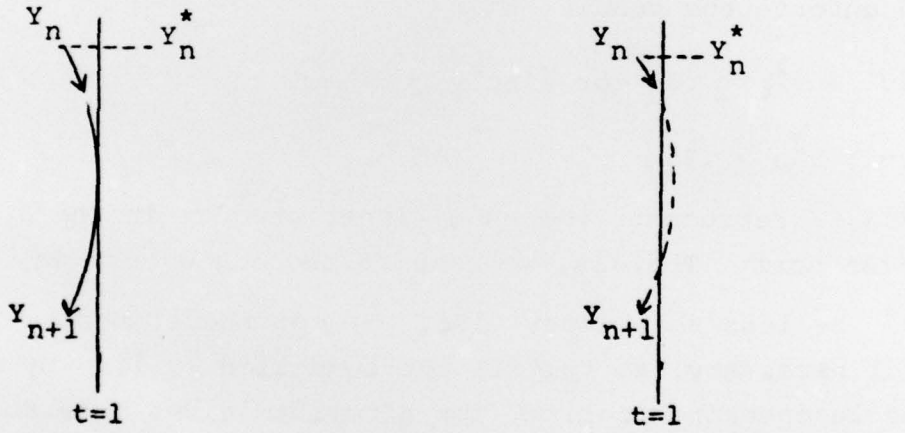
In general, it is preferable to let step size adjust automatically according to the smoothness of the curve. That is, the big "jump" is taken when the solution curve is "smooth" and small step is used otherwise. The flow chart does this by always doubling the step size used in taking the last step for use as the initial step size in the present step. Thus, if conditions (2.1), (3.1) do not provide any cut for the step size, it will grow exponentially after few steps. A

more efficient procedure than doubling every time is the following. Let Δs_i denote the step size implemented in the i^{th} step $i = 1,\cdots n$. Choose σ, the initial estimate for Δs_{n+1}, as follows:

1. If $n = 1$, then set $\sigma = 2\Delta s_1$.
2. If $n \geq 2$ and $\Delta s_n = 2\Delta s_{n-1}$, then start with $\sigma = 2\Delta s_n$.
3. If $n \geq 3$ and $\Delta s_n = \Delta s_{n-1} = \Delta s_{n-2}$ then start with $\sigma = 2\Delta s_n$.
4. Otherwise, we start with $\sigma = \Delta s_n$.

§5. Termination

When following the solution curve using the routine described in §2, 3, 4, one should also bear in mind the possibility that the path has changed the direction and possibly touched $t = 1$ as either of the pictures in the Figure.



Let F^T denote the t component of the vector field v with $H'v = 0$, i.e., $F^T = \frac{dT}{ds}$. Let $Y_n = (T_n, x_n)$, then the point symmetric to Y_n with respect to $t = 1$ is $Y_n^* = (2-T_n, x_n)$. Therefore, if $F^T(Y_n) > 0$ and $F^T(Y_{n+1}) < 0$ and if $|Y_n^* - Y_{n+1}| < \Delta s$, then the path may have "touched" or "passed over" $t = 1$. In this case we linearly interpolate F^T with $F^T(Y_n)$ and $F^T(Y_{n+1})$ as a function of s and find new $\Delta s'$ which gives approximately $F^T(\bar{Y}_{n+1}) \cong 0$, where $\bar{Y}_{n+1}$ represents the point on the path after Y_n with new step size $\Delta s'$.

In general, the path following routine will eventually carry the path to or pass through the region $R = \{(T,x) \mid (1-T) < \varepsilon\}$ where ε is some constant, usually choosen to be the estimated computational accuracy. We terminate the algorithm with (T_{n+1}, x_{n+1}) lies in R. If $T_{n+1} > 1$ and $(1-T_{n+1}) > \varepsilon$ we interpolate T as a function of s to find new step size Δs which gives approximately $T_{n+1} \geq 1$. Our experience shows that the quadratic interpolation using T_n, T_{n+1} and $T'_{n+1} = \frac{dT}{ds}\Big|_{n+1}$ is far more efficient then simpler linear interpolation on T_n and T_{n+1}. This should not be confused with quadratic interpolation using T_{n-1}, T_n, and T_{n+1} which may be quite bad.

§6. Remarks

The flow chart shown in §1 gives a general framework for the algorithm. Homotopy theory has given many situations in which the curve is guaranteed to end at T = 1. The algorithm may fail to follow such a curve to t = 1 for some rather singular situations. Although it is not listed in the Flow Chart, a program may wisely be designed to test for the following:

1. The rank of H' may become "nearly" less than n. In practice, it is impossible to demonstrate the rank is less than n, but certainly one might encounter a matrix (when following path) whose rank "appears" less than n. If we approach such a singular situation then the step size may become very small. A new choice of initial constants may remedy this situation, though such an occurrence suggests the problem has not been posed correctly.
2. c(s) may become unbounded.
   This is another type of difficulty that can occur if the homotopy is not in some sense well posed. The homotopy method can be used when one has no guarantee that path will continue to t = 1 when one is uncertain whether the path is advisible. Similarly, there is a third possibility.
3. A path may return to t = 0.
   If no check is made for such a possibility, considerable computation time may be wasted as the path wandering around in the region where t < 0.

Alden Wright, (Western Michigan University), has suggested (personal communication) that when faced with the possibility of $|c(s)| \to \infty$, it may sometimes be wise to rescale, and using new variables $(1, x_2/x_1, \cdots, x_n/x_1)$ and $\ell n|x_1|$ and $x_1/|x_1|$. He particularly recommends this change when dealing with systems of polynomial equations where the variables x_i are complex.

As we mention in the introduction, the emphasis on the Flow Chart is the "simplicity". The numerical results in the next section indicate that we can rely on this algorithm to carry the path to some t which is arbitrarily close to 1, as long as the machine precision permits. One may, of course, set $t = 1$ and switch to Newton method far before t reaches $|t-1| < \varepsilon$, where ε is the machine accuracy. The best strategy seems to be the following: Use Moore's Algorithm [15] on (T_n, x_n) if T_n is bigger than some value, say .9, to check if the Newton iteration on $F(x) = 0$ will converge by using x_n as starting point. We switch to Newton iteration if Moore's Algorithm indicates the convergence of Newton iteration. If Moore's Algorithm fails to guarantee the convergence of Newton iteration, we use the path following routine to push t one step further toward $t = 1$ and repeat the procedure. The implementation of this strategy is beyond the scope of the paper.

§7. Numerical results

A series of computer experiments were made, implementing the path following algorithm described in §2 - 4, for finding all the roots of polynomials of one complex variable. The first example is the Wilkinson polynomial [15] of degree 20. That is,

$$p(z) = (z+1)(z+2)\cdots(z+20) + z^{19}2^{-23} .$$

To solve $p(z) = 0$, we write

$$H(z,t) = (1-t)^k(z-a_1)\cdots(z-a_n) + (1-(1-t)^k)p(z)$$

where $a_i \in c$ for all $1 \le i \le n$. It can be shown that for almost all $a \equiv (a_1 \cdots a_n) \in c^n$, $H^{-1}(0)$ consists of n smooth paths all of them are bounded and connected to $t = 1$. So, we choose $(a_1, \cdots, a_n) \in c^n$ "at random", and follow each path individually to $|t-1| < \varepsilon$ with $\varepsilon = 10^{-12}$ and $k = 4$.

(The computations are done on Univac 1108 with complex double precision.) The results are shown on Table 1. As one can easily see that no root is missed and the step sizes varied from 10^{-8} to 6.4. $H(T_n, x_n)$ is sometimes as big as 10^{20} for points ε away from the curve, but this causes no difficulty. We evaluate the polynomial as a product of 20 terms, thereby avoiding one of the difficulties of roundoff error that Wilkinson emphasized. Nonetheless the polynomial is quite sensitive in the sense that the paths must be followed quite closely if they are going to be followed reliably.

The second example is the polynomials with multiple roots. Let

$$P_0(z) = (x+1)^3$$

and

$$H_0(t,z) = (1-t)^k(z-a_1)(z-a_2)(z-a_3) + (1-(1-t)^k)P_0(z) .$$

Table 2 shows that $P_0(z)$ does not cause difficulty on our algorithm.

To demonstrate the stability of our algorithm, we write

$$P_\varepsilon(z) = (x+1)(x+1+\varepsilon)(x+1-\varepsilon)$$

and

$$H_\varepsilon(z) = (1-t)^k(z-a_1)(z-a_2)(z-a_3) + (1-(1-t)^k)P_\varepsilon(z) .$$

Tables 3, 4, 5, 6 show the computation results for $\varepsilon = 10^{-1}$, 10^{-2}, 10^{-3} and 10^{-5}. Once again, it is shown that no roots are missed. We also show on Tables 8, 9, 10, 11 that the algorithm without angle checking is equally reliable in cases that solution path are "rather" smooth. While we are finding roots of a polynomial that has only real roots, an additional difficulty would occur if the roots were triple (or nearly triple) non-real roots since our algorithm then does not handle real roots in any special way.

Path	Starting point	Termination point	Biggest step size	Smallest step size	Number of Iterations	Total derivative Evaluations
1	(.490896+01, .827034+01)	(-.167307+02, .281262+01)	1.6	.820-08	85	832
2	(-.386351+01,-.640670+01)	(-.139924+02,-.251883+01)	.86	.296-04	42	507
3	(.227257+01,-.382116+01)	(-.300000+01,-.293843-025)	.8	.223-01	25	297
4	(-.560205+01,-.143901+01)	(-.600001+01, .645187-032)	.4	.362-02	26	319
5	(.165876+01, .351647+01)	(-.400000+01, .233519-032)	.8	.123-01	31	361
6	(.812979+00, .644600+01)	(-.100953+02, .643501+00)	.8	.229-08	40	461
7	(-.514428+00, .225853+00)	(-.100000+01,-.150951-19)	.4	.825-01	21	217
8.	(-.470877+01,-.346672+01)	(-.800727+01,-.605469-022)	.8	.381-03	20	259
9.	(-.113227+01,-.648676+01)	(-.500000+01,-.102315-033)	1.6	.656-03	29	331
10.	(.751382+00, .527626+01)	(-.699970+01, .940395-037)	1.6	.139-02	26	305
11.	(-.375093+01,-.426783+01)	(-.100953+02,-.643501+00)	.93	.934-08	38	427
12.	(.491118+01, .831259+01)	(-.195024+02, .194033+01)	3.2	.102-06	81	834
13.	(-.306077+01, .884518+01)	(-.139924+02, .251883+01)	1.6	.103-06	47	514
14.	(.705852+01,-.788806+01)	(-.167307+02,-.281262+01)	3.2	.232-08	58	628
15.	(-.687322+01,-.659123+01)	(-.117936+02,-.165233+01)	1.03	.266-04	39	440
16.	(-.123342+01, .556483+01)	(-.891725+01, .660052-41)	.8	.424-08	36	396
17.	(.173188+01, .490580+01)	(-.200000+01,-.429273-21)	1.6	.500-01	28	324
18.	(.821022+01,-.500563+01)	(-.117936+02, .165233+01)	2.04	.361-04	60	653
19.	(.989296+01, .796627+01)	(-.208469+02,-.351482-32)	6.4	.425-04	71	763
20.	(.935931+01,-.217296+01)	(-.19502439+02,-.194033+01)	6.4	.330-04	78	805

Table 1.

Path	Starting point	Termination point	Number of Iterations	Total derivative Evaluations
1	(.490896+01, .827034+01)	(-.100000+01, .296696-10)	35	386
2	(-.386351+01,-.640670+01)	(-.100000+01, .323699-10)	34	375
3	(.227257+01,-.382116+01)	(-.100000+01, .115901-09)	31	351

Table 2.

Path	Starting point	Termination point	Number of Iterations	Total derivative Evaluations
1	(.490896+01, .827034+01)	(-.100000+01, .236754-41)	22	233
2	(-.386351+01,-.640670+01)	(-.110000+01,-.159285-28)	21	223
3	(.227257+01,-.382116+01)	(-.900000+0 ,-.343765-28)	21	216

Table 3.

Path	Starting point	Termination point	Number of Iterations	Total derivative Evaluations
1	(.490896+01, .827034+01)	(-.100000+01, .507928-35)	23	241
2	(-.386351+01,-.640670+01)	(-.101000+01, .998031-23)	24	245
3	(.227257+01,-.382116+01)	(-.990000+0 ,-.720144-25)	22	237

Table 4.

Path	Starting point	Termination point	Number of Iterations	Total derivative Evaluations
1	(.490896+01, .827034+01)	(-.100000+01, .273186-40)	26	261
2	(-.386351+01,-.640670+01)	(-.100100+01,-.697884-37)	27	268
3	(.227257+01,-.382116+01)	(.999+0 , .143913-35)	24	242

Table 5.

Path	Starting point	Termination point	Number of Iterations	Total derivative Evaluations
1	(.490896+01, .827034+01)	(-.100000+01, .334193-51)	27	272
2	(-.386351+01,-.640670+01)	(-.100001+01,-.121369-30)	26	255
3	(.227257+01,-.382116+01)	(-.999990+0 ,-.242783-29)	26	262

Table 6.

Path	Starting point	Termination point	Number of Iterations	Total derivative Evaluations
1	(.490896+01, .827034+01)	(-.100000+01,-.470215-37)	15	144
2	(-.386351+01,-.640670+01)	(-.110000+01, .341001-19)	13	151
3	(.227257+01,-.382116+01)	(-.900000+0 ,-.113902-24)	12	163

Table 7.

Path	Starting point	Termination point	Number of Iterations	Total derivative Evaluations
1	(.490896+01, .827034+01)	(-.100000+01, .546147-39)	18	184
2	(-.386351+01,-.640670+01)	(-.101000+01,-.699258-21)	16	178
3	(.227257+01,-.382116+01)	(-.990000+0 , .111414-22)	15	182

Table 8.

Path	Starting point	Termination point	Number of Iterations	Total derivative Evaluations
1	(.490896+01, .827034+01)	(-.100000+01, .940167-37)	19	198
2	(-.386351+01,-.640670+01)	(-.100100+01, .162825-41)	20	198
3	(.227257+01,-.382116+01)	(-.999000+0 ,-.673943-18)	17	194

Table 9.

Path	Starting point	Termination point	Number of Iterations	Total derivative Evaluations
1	(.490896+01, .827034+01)	(-.100000+01, .343001-51)	27	272
2	(-.386351+01,-.640670+01)	(-.100001+1 ,-.121369-30)	26	255
3	(.227257+01,-.382116+01)	(-.999990+0 ,-.242783-39)	26	262

Table 10.

§8. Continuation versus Newton Methods

We have chosen to find roots of polynomials for this paper because polynomial root determination yields and interesting set of highly nonlinear test problems. We make no claim that continuation is an efficient way of finding roots of polynomials. We do not make use of the special "field" properties and in particular, after we find one root, we do not in any sense divide it out or even take it into consideration in any way for finding the other roots. We do not make any special search for real roots, (and having the roots real is not even a help). These practices make polynomials a reasonable set of test problems since in most nonlinear problems, such as finding roots of systems of polynomials or for inverse eigenvalue problems, special techniques are not available. The existence of several roots in our problems allow us to test here whether we actually follow different paths correctly.

Given our practices we asked how Newton's method would compare. For the Wilkinson polynomial

$$\prod_{k=1}^{20} (x+k) + 2^{-23}x^{19}$$

we chose several roots. For each root x_0, we chose a starting point of the form

$$x_0^t = x_0 + t(1+i) \quad .$$

By experimenting with several values of t, we determined how far from the root the initial point x_0 could be and still have Newton's method give a sequence converging to the root x_0. We chose the direction $1+i$ in defining x_0^t as a reasonable representative direction; given a starting point x_0^t, the closest root x_0 might be any direction form x_0^t. To use Newton's method to find the foots, starting points could be selected at random, and we test in essence how close the starting point must be and still get convergence to that root.

x_0	Newton's Method Converges for $t =$	Newton's Method Does Not Converge $t =$
-16.71+2.81i	.3	.4
-16.71-2.81i	.2	.3
-19.50+1.94i	.3	.4
-19.50-1.94i	.2	.3

Table 11.

Table 11 shows that for some of the roots Newton's method is successful only when x_0^t is quite close. Even when x_0 is closer to x_0^t than any other toot, by a factor of 7, (as is the case of the entries in Table 10) Newton's method may fail to converge to x_0.

Often one is interested in finding only one root, but the calculations in Table 10 suggest the nature of Newton's method: local converge. When starting outside some ill-defined boundary, Newton's method will often produce a sequence which diverges. More work is needed to demonstrate in much greater detail the inadequacy of Newton's Method on a variety of problems when compared with continuation and/or simplicial methods.

References

1. Alexander, J. C. (1978), The additive inverse eigenvalue problem and topological degree, Proceeding of A.M.S.
2. Alexander, J. C. and J. A. Yorke (1978), The homotopy continuation method: Numerically implementable topological procedures, Trans. Amer. Math. Soc.
3. Chow, S. N., J. Mallet-Paret and J. A. Yorke (1978), Finding zeros of maps: Homotopy methods that are constructive with probability one, Math. Comp., 32 (1978), pp. 887-899.
4. Chow, S. N., J. Mallet-Paret and J. A. Yorke (1978), "A homotopy method for locating all zeros of a system of polynomials", to appear.
5. Drexler, F. J. (1977), Eine Methode zur Berechnung sämtlicher Lösungen von polynomgleichungssystemen, Numer. Math., 29, pp. 45-58.

6. Eaves, B. C. and R. Saigal (1972), Homotopies for computation of fixed points on unbounded regions, Math. Programming, 3, 2, pp. 225-237.
7. Eaves, B. C. (1972), Homotopies for computation of fixed points, Math. Programming, 3, 1, pp. 1-22.
8. Garcia, C. B. and W. I. Zangwill (1978), Global continuation methods for finding all solutions to polynomial systems of equations in n variables, to appear in Symposium on Extremal Methods and Systems Analysis.
9. Garcia, C. B. and T. Y. Li (1979), On the number of solutions to polynomial systems of equations, MRC Technical Summary Report #1951.
10. Kellogg, R. B., T. Y. Li and J. A. Yorke (1976), A constructive proof of the Brouwer fixed point theorem and computational results, SIAM J. Num. Anal., 4, pp.473-483.
11. Scarf, H. (1967), The approximation of fixed points of a continuous mapping, SIAM J. Appl. Math., 15, 5, pp.1328-1343.
12. Smale, S. (1976), A convergent process of price adjustment and global Newton methods, J. Math. Econ., 3, pp. 1-14.
13. Watson, L. T., A globally convergent algorithm for computing fixed points of C^2 maps, Appl. Math. Comput., to appear.
14. Menzel, R. and H. Schwetlick, Zur Lösung parameterabhängiger nichtlinearer Gleichungen mit singulären Tacobi-Matrizen, Numer. Math. 30, pp. 65-79.
15. Moore, R. E., A computational test for convergence of iterative methods for nonlinear systems, SIAM J. Numer. Anal., Vol. 15, No. 6, pp. 1194-1196.

TYL/JAY/jvs

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: #1984

2. GOVT ACCESSION NO.

3. RECIPIENT'S CATALOG NUMBER

4. TITLE (and Subtitle): A Simple Reliable Numerical Algorithm for Following Homotopy Paths,

5. TYPE OF REPORT & PERIOD COVERED: Summary Report - no specific reporting period

6. PERFORMING ORG. REPORT NUMBER

7. AUTHOR(s): Tien-Yien Li [redacted] James A. Yorke

8. CONTRACT OR GRANT NUMBER(s): MCS28-02420; MCS78-09525; DAAG29-75-C-0024; NSF-MCS76-24432

9. PERFORMING ORGANIZATION NAME AND ADDRESS: Mathematics Research Center, University of Wisconsin
610 Walnut Street
Madison, Wisconsin 53706

10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: Work Unit #2-Other Mathematical Methods; Work Unit #5-Math. Programming & Operations Research.

11. CONTROLLING OFFICE NAME AND ADDRESS: See Item 18 below

12. REPORT DATE: Jul[redacted]79

13. NUMBER OF PAGES: 18

14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office): Technical summary rept.,

15. SECURITY CLASS. (of this report): UNCLASSIFIED

15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

MRC-TSR-1984

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

U. S. Army Research Office
P. O. Box 12211
Research Triangle Park
North Carolina 27709

and

National Science Foundation
Washington, D. C. 20550

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Homotopy, path-following, continuation methods.

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

The purpose of this paper is to develop simple algorithms and flow charts for the homotopy continuation methods so that scientists may easily program reliable curve followers. The emphasis is on simplicity and reliability at the expense of the speed.

22.1 2.00

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)