Pergamon

Computers Math. Applic. Vol. 27, No. 7, pp. 1–5, 1994
Copyright©1994 Elsevier Science Ltd
Printed in Great Britain. All rights reserved
0898-1221/94 $6.00 + 0.00

0898-1221(94)E0020-K

Approximate Root Isolation for Nonlinear Systems by Monte-Carlo

B. JONES AND S. IYENGAR
Computer Science Department, Louisiana State University
Baton Rouge, LA 70803-4020, U.S.A.

(Received September 1992; accepted November 1993)

Abstract—A procedure for isolating the real roots of nonlinear systems of equations is presented. The technique requires only function values and can be applied readily to complicated systems of transcendental functions. A program implementing the technique has been written, and the effectiveness and efficiency of the program are discussed.

Keywords—Monte-Carlo, Nonlinear systems, Root isolation, Chebyshev.

INTRODUCTION

A technique to isolate the real roots of nonlinear systems of equations is presented, and a program implementing the technique is discussed. The procedure generalizes methodology for a single transcendental equation [1,2]. When only a single root is needed, the quickest method, provided it works, is an iterative technique [3]. If an iterative technique fails, or if it is desirable to isolate more roots, the method proposed here might be tried—it has worked in all instances where it has been tried. Grid techniques often miss roots and involve an exponential increase in points with dimensionality [4]. The technique we propose provides much greater certainty, and avoids, except as a worst case, exponential explosions. Also, many techniques require the same number of equations as variables—a restriction we do not impose.

Given the set of nonlinear equations

$$\begin{aligned} f_1(x_1, x_2, \dots, x_n) &= 0, \\ f_2(x_1, x_2, \dots, x_n) &= 0, \\ &\vdots \\ f_m(x_1, x_2, \dots, x_n) &= 0, \end{aligned}$$

and an initial region described as

$$a_i \le x_i \le b_i, \qquad i = 1, 2, \dots, n,$$

by employing only function values, we attempt to enclose the roots of the system in disjoint regions.

$$R_j = (\ell_{ji}, U_{ji}) \qquad i = 1, 2, \dots, n, \quad j = 1, 2, \dots, p,$$

This work was supported by the Office of Naval Research under Grant NOOO14-91-J-1306.

Typeset by $\mathcal{A}\mathcal{M}\mathcal{S}$-TEX

where the i^{th} two tuple bounds the i^{th} coordinate. Our intention is that each region be small according to a prescribed tolerance on $U_{ji} - \ell_{ji}$, and each region contain one root. However, since the system might sometimes be zero over some type of region, we provide for a larger than prescribed final region. Further, we allow $m = n, m < n$ or $m > n$.

No algorithm which uses only function values can be guaranteed to isolate roots, since only a finite number of points can be employed. Moreover, since a prescribed tolerance on terminal regions is employed, more than one root may exist in a terminal region. Thus, the problem is deterministically unsolvable; however, we can give algorithms which isolate roots while characterizing the functions for which they might fail. By forcing the class of functions for which the algorithms might fail to have "extremes" of behavior, we gain "confidence" that we have isolated the roots.

METHODOLOGY

To facilitate understanding of the algorithm, a concise treatment of algorithm constituency is first presented. We introduce the following simplified algorithm abstraction.

```
(a) Initial region → $R$
(b) Test ($R$): Negative, delete $R$ and Proceed to $c$.
              : Positive, Partition ($R$) into $r_1, r_2$. Stack $r_1, r_2$.
(c) Pop stack → $R$ (if empty go to $e$.)
(d) Size ($R$) < tolerance: No, go to $b$
              : Yes, $R$ → outstack, go to $c$.
(e) Return region in outstack.
```

The law of large numbers and the fact that a region is deleted or reduced to smaller regions (each smaller than the previous region by at least a fixed region size) can be invoked to show the algorithm terminates. For a feel for efficiency, we rely on empirical analysis.

The constituent Test (R) yields a positive indication if

$$\frac{\bar{f}_j^2}{V_j} \leq K^2, \qquad j = 1, 2, \ldots, m,$$

where $\bar{f}_j$ is the mean of the j^{th} function over R, V_j is the function variance and K^2 is a prescribed constant. Test (R) iteratively doubles the number of random points in R until each $\bar{f}_j^2/V_j$ roughly converges. If the mean squared over variance test is not satisfied for every function, we have a negative indication. The basis for the mean squared over variance test is Chebyshev's inequality and empirical analysis. Chebyshev's inequality tells us that

$$P\left\{|x - \mu| \geq k\sigma\right\} \leq \frac{1}{k^2}.$$

This holds, regardless of the distribution of the random variable x, where μ is the mean and σ is the standard deviation. In our analysis, we take $f_j(x_1, \ldots x_n)$ as our random variable, and we use the sample mean $\bar{f}_j$ and sample variance V_j as estimates of μ and σ^2. The formula we use for the mean yields a consistent unbiased estimate of μ, and the weak law of large numbers guarantees that we will converge to μ [5]. The formula we use for the standard deviation yields the maximum probability estimate of σ [5]. The problem of estimating σ is alleviated because only a rough estimate is needed; as we demonstrate below, Chebyshev's inequality is not sharp. Although we have not employed variance reduction techniques in our program, this is another possibility which might also reduce the amount of computation which must be done. In this regard, a discussion of stratified sampling and Russian roulette can be found in [6].

Chebyshev's inequality states that the amount of area under any distribution curve, which is further away from the mean than k standard deviation units, is less than $1/k^2$. For symmetrical distributions with a single mode, the maximum area farther than $k\sigma$ from the mean is $.444/k^2$.

A comparison of the actual and Chebyshev-maximum area in the tails beyond $\mu - k\sigma$ and $\mu + k\sigma$ is given for a number of distributions in Table 1. If the number n of degrees of freedom is greater than 2, the t distribution has mean 0 and variance $\frac{v}{(v-2)}$. The χ^2 distribution with v degrees of freedom has mean v and variance $2v$. For small values of v, the χ^2 distribution is so skewed that the mean is less than one standard deviation above zero, and consequently, for this distribution, only the values marked with an asterisk actually represent two-tail areas.

The inequality may be interpreted here as a bound on the percentage of values of the function outside a certain range. The percentage of values, if any, outside the range is usually much smaller, since Chebyshev's inequality is not sharp. If the mean squared over variance test is not satisfied for a large value of K, it is very unlikely that the function has a zero in the region. Choices of K are further tempered by empirical analysis.

Our objectives in Partition (R) are to quickly close on roots in an uncomplex manner and avoid a combinatorial explosion of regions when many variables are involved. We chose to accomplish these objectives by bisecting the bounds in one coordinate each time Partition (R) is executed. Each time Partition (R) is reached, the next successive coordinate is selected, cycling back to the first coordinate from the last. The behavior of the functions determines the number of regions which must be examined to determine there are not other roots. It is possible for a system to require more regions to be examined than a somewhat larger system.

Table 1. Tail areas beyond $\mu - k\sigma$ and $\mu + k\sigma$ for several distributions.

k	t Distribution		χ^2		Normal	Chebyshev Max. Tail Area	
	$v = 3$	$v = 10$	$v = 1$	$v = 10$		*Any*	*Symmetric*
1	.182	.290	.121	.297*	.317	1.000	.444
2	.020	.050	.050	.041*	.046	.250	.111
3	.015	.007	.022	.009	.003	.111	.049
4	.0064	.0011	.0099	.0019	.00006	.063	.028

The number of points for a region depends only on σ [6]. This number of points might be further reduced by variance reduction techniques [6]. While it is true that σ often increases with the dimensionality, there is no reason in principle why this should be so. The exponential increase seen in grid searches almost never occurs in Monte-Carlo [6]. However, in the worst case, the search degenerates to a grid type search. Our empirical analysis yields some insight into the effects of dimensionality.

The component size, $(R_j) <$ tolerance, checks the bounds on each coordinate

$$U_{ij} - \ell_{ij} < E, \qquad i = 1, 2, \ldots, n.$$

Sometimes our algorithm will produce a couple of regions which are extraneous that adjoin the root-containing region, or adjoining regions may be produced which are relevant, since the system may be zero over some type of region. For this reason, as regions are placed on the outstack, adjoining regions are consolidated. Terminal regions may thus be produced which are larger than the specified tolerance. With the extraneous case in mind, the user might choose the terminal tolerence slightly smaller than actually desired (e.g., $E = \frac{\varepsilon}{4}$, where ε is the desired size). If a region of zeros is suspected, a larger value of E might be employed to avoid numerous consolidations.

As the systems become more complex or larger, program run-time is dominated by system evaluations. The arithmetical operations required depends on the particular nonlinear system. Referring back to our algorithm abstraction, we see that Test (R) is executed, and the region is subsequently divided into two regions upon a positive indication. These two regions are later subjected to examination by Test (R). Now, Test (R) involves computation of system values, and examination of the subregions involves computation again of values in the region. As regions are repeatedly tested in this manner, regional recomputations become numerous. For this reason, dynamic list processing routines have been included to re-use function values in regions.

Initially, a free list is created, and subsequent list processing extracts or returns cells from this free pool. At the partition step, a new list is created, and appropriate values are withdrawn from the previous list and inserted in this list as they are required. If there are insufficient values in the previous list corresponding to this region, then new values are computed and added to the list. When a region is deleted, the corresponding list is freed. Empirically, the list processing usually eliminates fifty percent of the system evaluations.

EMPIRICAL DEMONSTRATION

The ROOT program has been successfully employed on numerous nonlinear systems. In all test cases, the program correctly isolated the roots; however, in some cases it was necessary to increase the parameter TK (the value of K^2), until all roots were found. The parameter TK is the crucial parameter in root isolation, and we temper the choice of TK with this empirical analysis. The test systems had from two to six variables, two to six equations, and varied roots. Various regions and parameter settings were employed to determine effectiveness and efficiency. Based on this empirical analysis, we make some observations and suggestions for the program. We note that in the following examples cpu time is given for runs on a 66 MHz 486DX2 computer with 8 MB RAM after compilation by the Microsoft FORTRAN power station. The results below typify the results obtained, so it is not necessary to reproduce a large number of examples.

Before generalizing, we exhibit some typical cases. Consider the system

$$\begin{aligned} x_1 - \sin(x_1 + x_2) + x_3 - 1 &= 0, \\ (x_2 - \cos(x_1 - x_2))^2 + (x_3 - 1)^2 &= 0, \end{aligned}$$

with each variable bounded between 0 and 4. With $TK = 7$ and terminal tolerance $E = .01$ the program correctly isolated the root in 35 seconds of cpu time.

Next, we consider the system

$$\begin{aligned} x_1 - x_2^2 + 2 &= 0, \\ -x_1^2 + x_2 + x_3 &= 0, \\ 2x_1 - x_2^2 + x_3 &= 0, \end{aligned}$$

with each variable bounded between -10 and 10. There are four roots in this region, and the program correctly isolated them with $TK = 7$. With the terminal tolerance $E = .01$, the program ran in under two minutes of cpu time.

The following system is almost linear.

$$\begin{aligned} 2x_1 + 2x_2 + 3x_3 + 2x_4 + 2 &= 0, \\ 4x_1 - 3x_2 + x_4 + 7 &= 0, \\ 6x_1 + x_2 - 6x_3 - 5x_4 - 6 &= 0, \\ -8x_1 + x_2 e^{x_3} + x_4^2 + 4x_4 - e^{x_3} &= 0. \end{aligned}$$

Each variable is bounded between -5 and 10. The root was correctly isolated with $TK = 6$ and an $E = .01$ in under three minutes of cpu time.

Jumping to a six by six system, we consider

$$\begin{aligned} 2x_1 - x_2 + 4x_3 - 3x_4 + x_5 - 11 &= 0, \\ -x_1 + x_2 + 2x_3 + x_4 + 3x_5 - 7x_6 &= 0, \\ 4x_1 + 2x_2 + 3x_3 + 3x_4 - x_5 - x_6^2 &= 0, \end{aligned}$$

$$\begin{aligned}
-3x_1 + x_2 + 3x_3 + 2x_4 + 4x_5 - 2x_6^3 &= 0,\\
x_1 + 3x_2 - x_3 + 4x_4 + 4x_5 - 9x_6 &= 0,\\
-4x_1 + 2x_2x_4 + x_3e^{x_2} - e^{x_2} + x_5^2 - 2x_5 &= 0,
\end{aligned}$$

with $E = .01, TK = 6$, and each variable bounded between -3 and 6. Two roots were isolated in approximately seven minutes.

These examples give a feel for efficiency. Generally, there is an increase in cpu time with an increase in variables, an increase in region, an increase in TK, an increase in roots, or a decrease in the terminal tolerance. However, the increase is dependent on the behavior of the functions. For example, in our test cases some three by three systems with $TK = 9$ required less run time than three by three systems with $TK = 6$. For the same value of TK, we had five by five systems requiring less run time than three by three systems. There is no exponential explosion associated with these factors for our technique as there is for a grid type approach. As far as program parameters are concerned, on the same system, incrementing TK by 1 commonly increases computation by 25%, and decreasing E from .01 to .001 often increases computation by 30%.

From our empirical analysis and consideration of Chebyshev's inequality, we suggest a TK of 7 to 9 in an unfamiliar situation. If the user develops some experience with a class of problems, a less restrictive choice for TK may be found to suffice.

CONCLUDING REMARKS

The 486DX2 employed in these examples is a desktop computer, and is not a fast computer by current standards. It does serve to give us a feel for how computation time increases with different parameterizations. The storage requirements are minimal, and the technique runs on any computer with 8 MB RAM or more. With current machine speeds and availabilities, it is feasible to solve much larger nonlinear systems with this approach. This approach is also amenable to parallel computation.

REFERENCES

1. B. Jones, W. Waller and A. Feldman, Root isolation using function values, *BIT* **18**, 311 (1978).
2. B. Jones, A heuristic for root isolation, *Int. J. Num. Meth. Eng.* **11**, 598 (1977).
3. J.M. Ortega and W.C. Rheinbolt, *Iterative Solution of Nonlinear Equations in Several Variables*, Academic Press, New York, (1970).
4. W. Forster, *Numerical Solution of Highly Nonlinear Problems*, North-Holland, New York, (1980).
5. J.F. Kenny and E.S. Keeping, *Mathematics of Statistics*, D. Van Nostrand, New York, (1963).
6. H. Kahn, Multiple quadrature by Monte-Carlo methods, In *Mathematical Methods for Digital Computers*, (Edited by A. Ralston and H. Wilf), Volume 1, Wiley, New York, (1967).