ALGORITHM 681
INTBIS, a Portable Interval Newton/Bisection Package

R. BAKER KEARFOTT AND MANUEL NOVOA III
University of Southwestern Louisiana

We present a portable software package for finding all real roots of a system of nonlinear equations within a region defined by bounds on the variables. Where practical, the package should find all roots with mathematical certainty. Though based on interval Newton methods, it is self-contained. It allows various control and output options and does not require programming if the equations are polynomials; it is structured for further algorithmic research. Its practicality does not depend in a simple way on the dimension of the system or on the degree of nonlinearity.

Categories and Subject Descriptors: G.1.5 [**Numerical Analysis**]: Roots of Nonlinear Equations—*systems of equations*

General Terms: Algorithms, Design, Reliability, Verification

Additional Key Words and Phrases: All solutions, generalized bisection, global constrained optimization, interval arithmetic, interval Newton methods

1. INTRODUCTION AND PURPOSE

Interval Newton methods in conjunction with generalized bisection are one approach which can be used to find all solutions to a system of nonlinear equations. Specifically, it can find, *with certainty*, approximations to all solutions of the nonlinear system:

$$f_i(x_1, x_2, \ldots, x_n) = 0, \qquad 1 \le i \le n, \tag{1.1}$$

where bounds a_i and b_i are known such that:

$$a_j \le x_j \le b_j \qquad \text{for} \quad 1 \le j \le n.$$

This approach can, even when implemented in machine arithmetic, obtain results for (1.1) with *mathematical certainty*. (See, e.g., [10] for a discussion of this; see [1] or [12] for an introduction to the concepts, and [13] for recent reviews and references.)

Authors' address: Department of Mathematics and Statistics, University of Southwestern Louisiana, Lafayette, LA 70504-1010.

Drawbacks to interval Newton methods in general include lack of universal hardware and software support for interval arithmetic. The INTBIS package, however, is self-contained, conforms to the ANSI 1977 FORTRAN standard, and contains thorough in-line documentation. Furthermore, its output formats and other aspects of the algorithm can be controlled without recompiling, and no programming is required when the functions f_i are polynomials. Users do not need a knowledge of interval arithmetic. Nonetheless, INTBIS is sufficiently modularized and documented to be useful as a test bed for new algorithmic ideas. INTBIS also stores and (optionally) prints various quantities useful for tracing algorithmic performance.

2. A BRIEF DESCRIPTION

The principal subroutine is structured on Algorithm 2.5 in [6] (see also [11]). The entire package is similar to the code described in [7], with the following differences.

(1) We use simulated directed roundings in the interval arithmetic. The installer provides a rigorous estimate for the number of units in the last place by which the stored result of one of the four elementary arithmetic operations or exponentiation can be in error. The package then produces interval results that contain the true results. The effect is a relatively simple and transportable, yet rigorous, interval arithmetic. With such a scheme, INTBIS can be guaranteed not to miss any roots.

(2) We use the interval Gauss–Seidel method with extended interval arithmetic (cf. [4] and [5]) in lieu of the Krawczyk method in [7]. The former is usually more efficient and flexible.

(3) We need to evaluate the components f_i at points in R^n during the interval Gauss–Seidel process. We have found it necessary to use interval arithmetic with our simulated directed roundings to do these computations; otherwise, roundoff errors would occasionally cause INTBIS to fail to find roots in the initial region.

(4) We have changed the criteria for deciding whether to iterate the interval Newton (i.e., interval Gauss–Seidel) process or to bisect a coordinate interval instead. We now examine the ratio of volumes of the original region to the region after application of the interval Gauss–Seidel step, where we ignore dimensions for which the box widths are smaller than the stopping diameter. (Note that this only affects division of the box into smaller boxes in which Newton's method is guaranteed to converge. Once such smaller boxes are obtained, the classical Newton's method is used to get accurate point approximations to the roots.)

(5) We use a more efficient scheme to choose which coordinate interval to bisect. In particular, let X be the box defined by the inequalities in (1.1), let $f_i(X)$ be an interval extension of f_i evaluated at X, and let $F(X)$ be the interval column vector whose elements are the $f_i(X)$. Let $J(X)$ be a corresponding interval extension of the Jacobian matrix, and let $[J_{i,j,1}, J_{i,j,2}]$ be the interval in the (i, j)th entry. Then, in contrast to the scheme in Section 2 of [7], we define

$$s_j = \max_{1 \le i \le n} \{ |J_{i,j,1}|, |J_{i,j,2}| \}(b_j - a_j).$$

We then bisect in the coordinate with index j for which s_j is maximum. This coordinate direction is, roughly, the one in which the values of the f_i change most rapidly; its choice for bisection introduces an implicit scaling.

We note that INTBIS is sufficiently modularized to allow interval Newton methods, other than the interval Gauss–Seidel, and to allow alternate coordinate selection schemes, with modest modifications.

In designing INTBIS, we have balanced simplicity, clarity, and ease of use and modification against efficiency.

3. STRUCTURE

There are 38 routines, which occupy approximately 250 kilobytes; over half of these files consists of comments. The package is organized into

(i) the primary package;
(ii) the interval arithmetic subpackage; and
(iii) the stack and linked list management subpackage.

Additionally, the package references modules from LINPACK, see [2]. The package also requires the routine D1MACH to obtain floating point machine constants. This routine is part of the SLATEC library, and is also available from NETLIB, see [3].

The interval arithmetic subpackage includes subroutines ADD, MULT, POWER, RNDOUT, SCLADD, SCLMLT, SIMINI, SUB, XDIV, XINT, and XSCLSB. Users may replace these with machine-specific ones if they so desire.

The stack management subpackage includes subroutines ADDBOX, ALLOC, DELBOX, FREE, POP, and PUSH.

The driver routine is GENBIS, where array storage is allocated. Input occurs mainly in INPUT, whereas output occurs mainly in OUTPUT and ERROUT. (The only other I/O is in GENBIS.)

Subroutine ROOTS embodies the main algorithm, whereas subroutine FTESTH indexes storage for the interval Gauss–Seidel routine HNSNG and for the coordinate selection routine PVSLCT. The routine HNSNG calls POLFUN for interval function values and calls POLJAC for interval Jacobian matrix values.

Subroutine OUTPUT calls routine NEWTON, which performs the classical Newton method. Subroutine NEWTON in turn calls POLFSC and POLJSC, which do noninterval function and Jacobian matrix evaluations.

See Figure 1 for a calling diagram for the major routines.

The routines POLFUN, POLJAC, POLFSC, and POLJSC work with polynomial systems in a general format. However, the interval arithmetic subpackage contains the essential elements for writing both more efficient and specialized routines and for writing routines that involve evaluation of more general transcendental functions. Otherwise, only GENBIS need be modified, since these subroutine names are passed as arguments.

The first author is developing portable FORTRAN-77 interval routines for the elementary transcendental functions, which can be used with GENBIS with or without the aid of a precompiler. Including such routines in INTBIS in a way

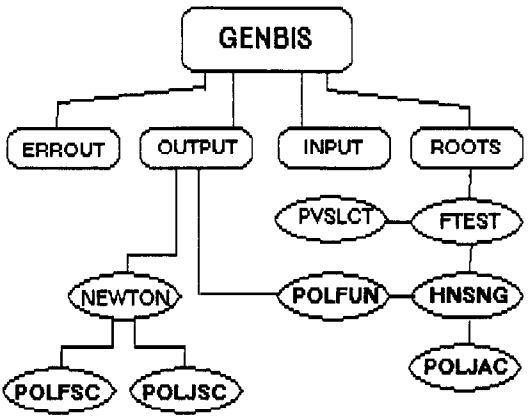

Fig. 1. Main structure and routines in INTBIS.

that does not require programming would make the package substantially larger and more complicated. However, the first author will send these routines and advice on their use upon request.

Also, ambitious users may easily supply their own coordinate selection scheme and interval Newton method, since these functions are isolated in the external routine FTESTH, which the driver routine GENBIS passes as an argument.

Another routine of interest is DIAMCP; the scaled distance between two numbers in the domain of F is computed there. Users may wish to alter the scaling.

Additional auxiliary routines in INTBIS include BISECT, CHKLST, DELLST, DVSBIN, ERRHND, EXPAND, and XINFO.

4. INSTALLATION

Installation involves

(i) making sure input and output units are correct and possibly supplying appropriate attachments;
(ii) possibly adjusting the size of workspace parameters;
(iii) possibly setting machine-dependent constants; and
(iv) making sure the package has access to the appropriate LINPACK routines.

Doing this involves making only several small changes in the code in well-marked and isolated places. Complete details appear in a file that is distributed with the package.

5. USE

INTBIS may be used by itself, or an alternate driver routine to GENBIS may be supplied. Output is controlled with a flag and ranges from no output to debugging and performance related output. Another flag selects four possible formats for output of floating point vectors.

Several flags also control variation of the algorithm. For example, expansion and deletion steps as described in [6] and [7] may be omitted. Such steps help to reduce redundancy and increase efficiency when roots occur on boundaries, but may cause unusual behavior with singular roots. The algorithm also optionally retains the regions such steps have rejected.

The user sets control flags, tolerances, and a bound on the total amount of work in the configuration file. The file CONFIG.BIS is a sample file which can be used as a template; see the documentation in INPUT.

The package is set up to solve polynomial systems of equations (see Section 3, above). We use tableau input to describe such a system; the portion of our input file associated with the equations has exactly the same format as that for the routine CONSOL8 in [14]. An example file (usable as a template) appears in INPUT; also see the file TOMS86AL.DT1.

If INTBIS is embedded in a larger code, it is probably easiest to use the supplied driver GENBIS as a template for the program that calls ROOTS. In such contexts, a call to OUTPUT after the call to ROOTS will determine the number of roots NINLST and will produce a pointer array INDBOX which gives addresses of the roots; see the documentation in the header to OUTPUT.

Without modification, INTBIS should perform well on many small systems of equations. If property installed, the only way INTBIS should fail is by not completing within the bounds on the amount of work or storage. A larger number of equations and variables does not necessarily imply a larger amount of work, but the precise amount of work seems less predictable in larger dimensions; see [10] and the references therein for a discussion of this. Systems that combine high nonlinearity, strong coupling, and a large number of terms pose more of a problem. For such systems, INTBIS may not be practical with the default routines POLFUN and POLJAC, but may work better if these are replaced; see [15]. Also, other schemes are available for systems with patterns in the nonlinearity or singularities; see [8] and [9].

Finally, we point out that the software interval arithmetic in INTBIS is slower than the best low-level implementations, where such are available. With good compiler support for interval arithmetic, the user can replace subroutines HNSNG, POLFUN, and POLJAC to effect a substantial speedup.

BIBLIOGRAPHY

1. ALEFELD, G., AND HERZBERGER, J. *Introduction to Interval Computations.* Academic Press, New York, 1983.
2. DONGARRA, J. J., MOLER, C. B., BUNCH, J. R., AND STEWART, G. W. *LINPACK Users' Guide.* SIAM, Philadelphia, 1979.
3. DONGARRA, J. J., AND GROSSE, E. Distribution of mathematical software via electronic mail. *ACM SIGNUM Newsl. 20,* 3 (July 1985), 45–47.
4. HANSEN, E. R., AND GREENBERG, R. I. An Interval Newton method. *Appl. Math. Comput. 12* (1983), 89–98.
5. HANSEN, E. R., AND SENGUPTA, S. Bounding solutions of systems of equations using interval analysis. *BIT 21* (1981), 203–211.
6. KEARFOTT, R. B. Abstract generalized bisection and a cost bound. *Math. Comput. 49,* 179 (July 1987), 187–202.
7. KEARFOTT, R. B. Some tests of generalized bisection. *ACM Trans. Math. Softw. 13,* 3 (Sept. 1987).

8. KEARFOTT, R. B. On handling singular systems with interval Newton methods. In *Proceedings of the Twelfth IMACS World Congress on Scientific Computation*, 1988.
9. KEARFOTT, R. B. Preconditioners for the interval Gauss–Seidel method. *SIAM J. Numer. Anal. 27*, 3 (June 1990).
10. KEARFOTT, R. B. Interval arithmetic methods for nonlinear systems and nonlinear optimization: An introductory review. In *Impacts of Recent Computer Advances on Operations Research*. Elsevier, New York, 1989.
11. MOORE, R. E., AND JONES, S. T. Safe starting regions for iterative methods. *SIAM J. Numer. Anal. 14*, 6 (Dec. 1977), 1051–1065.
12. MOORE, R. E. *Methods and Applications of Interval Analysis*. SIAM, Philadelphia, 1979.
13. MOORE, R. E., ED. *Reliability in Computing*. Academic Press, New York, 1988.
14. MORGAN, A. P. *Solving Polynomial Systems using Continuation for Engineering and Scientific Problems*. Prentice-Hall, Englewood Cliffs, N.J., 1987.
15. RATSCHEK, H., AND ROKNE, J. G. *Computer Methods for the Range of Functions*. Horwood, Chichester, England, 1984.

Received August 1988; revised April 1989; accepted May 1989