INTERVAL MATHEMATICS TECHNIQUES FOR CONTROL THEORY COMPUTATIONS

R. Baker Kearfott

Department of Mathematics
University of Southwestern Louisiana
Lafayette, Louisiana 70504-1010

1 Introduction and motivation

Various types of nonlinear equations or systems of equations arise in elementary and advanced control theory. For example, the transfer function corresponding to a single linear controlled ordinary differential equation is a rational function (cf. eg., [2], Sect. 1.2.). Stability of such systems depends on whether there are any roots of the numerator and denominator of the transfer function in the right half of the complex plane.

Analytical techniques have been used to check this condition. However, as is indicated below, we may also check it computationally. In particular, interval methods can find, *with the rigor of a mathematical proof*, *even on a computer*, *all* roots of a function in the right half of the complex plane. Thus, if such a method fails to find any root, the condition is valid; conversely, if the method finds a single such root, then no more computation is required.

More generally, interval methods can be used to rigorously find all roots of a nonlinear system of equations within given bounds on the variables.

In optimal control, we may wish to find a control function which optimizes a certain functional subject to a condition. The condition is usually a differential equation which describes the behavior of the corresponding physical system. (See [2], Ch. 6.) Such problems have been approached via calculus of variations techniques. Alternately, we may discretize the equation and condition to transform the problem into a finite dimensional nonlinear optimization problem. In theory, interval methods can then be used to find *with mathematical rigor*, the *global* optima. We comment below on the practicality of this for objective functions with various properties.

We may also wish to check whether a certain quantity (such as an output or a control) lies within certain bounds. With interval arithmetic, we can *computationally but rigorously* determine bounds on the range of a function over a given domain.

This paper is meant to provide illustrative examples, to indicate where the methods may be practical in control theory, and to describe some present research directions. In Section 2, we introduce the elementary aspects of interval arithmetic, show how these can be used to rigorously bound the range of a function, and point to references. In Section 3, we briefly explain how interval arithmetic can solve nonlinear systems of equations and nonlinear optimization problems, and give references for further study. In Section 4, we propose interval arithmetic procedures to determine whether a polynomial has roots in the right half plane. In Section 5, we comment on the practicality of interval methods and we mention research which will expand their range of applicability.

2 How interval arithmetic works

Refer to [1] or [16] for a thorough introduction to interval mathematics. Further references appear in [11], and a list of about 2000 papers on the subject appears in [4] and [5]. The recent proceedings [17] contains assessments of the role of interval mathematics in scientific computation. Here, we will give an elementary explanation of some of the most important concepts.

We will denote interval quantities throughout with boldface.

Interval arithmetic is based on defining the four elementary arithmetic operations on intervals. Let $\mathbf{a} = [a_l, a_u]$ and $\mathbf{b} = [b_l, b_u]$ be intervals. Then, if $op \in \{+, -, *, /\}$, we define

$$\mathbf{a}\ op\ \ \mathbf{b} = \{x\ op\ \ y \mid x \in \mathbf{a} \text{ and } y \in \mathbf{b}\}\,. \tag{2.1}$$

For example, $\mathbf{a} + \mathbf{b} = [a_l + b_l, a_u + b_u]$. In fact, all four operations can be defined in terms of addition, subtraction, multiplication, and division of the endpoints of the intervals, although multiplication and division may require comparison of several results. The result of these operations is an interval except when we compute $\mathbf{a}/\mathbf{b}$ and $0 \in \mathbf{b}$. (See eg. [16], pp. 66-68 for a discussion of the latter case.)

A large part of interval mathematics' power lies in the ability to compute *inclusion monotonic interval extensions* of functions. If f is a continuous function of a real variable, then an inclusion monotonic interval extension $\mathbf{f}$ is defined to be a function from the set of intervals to the set of intervals, such that, if $\mathbf{x}$ is an interval in the domain of $\mathbf{f}$,

$$\{f(x) \mid x \in \mathbf{x}\} \subset \mathbf{f}(\mathbf{x})$$

and such that

$$\mathbf{x} \subset \mathbf{y} \implies \mathbf{f}(\mathbf{x}) \subset \mathbf{f}(\mathbf{y}).$$

Inclusion monotonic interval extensions of a polynomial may be obtained by simply replacing the dependent variable by an interval and by replacing the additions and multiplications by the corresponding interval operations. For example, if $p(x) = x^2 - 4$, then $\mathbf{p}([1, 2])$ may be defined by

$$\mathbf{p}([1, 2]) = ([1, 2])^2 - 4 = ([1, 4]) - [4, 4] = [-3, 0].$$

The result of an elementary interval operation is precisely the range of values that the usual result attains as we let the operands range over the two intervals. However, the value of an interval extension of a function is not precisely the range of the function over its interval operand, but only contains this range, and different (mathematically equivalent) forms of the function give rise to different interval extensions. For example, if we write p above as $p(x) = (x-2)(x+2)$, then the corresponding interval extension gives

$$\begin{aligned}\mathbf{p}([1,2]) &= ([1,2]-2)([1,2]+2) = [-1,0][3,4] \\ &= [-4,0],\end{aligned}$$

which is not as good as the previous extension.

We may use the mean value theorem or Taylor's theorem with remainder formula to obtain interval extensions of transcendental functions. For example, suppose $\mathbf{x}$ is an interval and $a \in \mathbf{x}$. Then, for any $y \in \mathbf{x}$, we have

$$\sin(y) = \sin(a) + (y-a)\cos(a) - (y-a)^2/2\sin(c)$$

for some c between a and y. If a and y are both within a range where the sine function is non-negative, then we obtain

$$\sin(y) \in \sin(a) + (\mathbf{x}-a)\cos(a) - \frac{(\mathbf{x}-a)^2}{2}.$$

The right side of this relationship gives the value of an interval extension of $\sin(x)$, albeit a somewhat crude one.

See [23] for more techniques of producing interval extensions.

Mathematically rigorous interval extensions can be computed in finite precision arithmetic via the use of *directed roundings.* If x and y are machine-representable numbers and op is one of the four elementary operations $+$, $-$, $*$, or $/$, then, $x \; op \; y$ is not normally representable in the machine's memory. In interval arithmetic with directed rounding, if $\mathbf{x} \; op \; \mathbf{y} = [c,d]$, then we always round the computed value for c down to a machine number less than the actual value of c, and and we always similarly round the value for d up. To be completely rigorous, we also first apply directed rounding to the initial data while storing it.

If interval arithmetic with directed rounding is used to compute an interval extension $\mathbf{f}$ of f, if $[c,d] = \mathbf{f}([a,b])$, and $[c,d]$ does not contain zero, then this is a rigorous proof (regardless of the machine wordlength, etc.) that there is no root of f in $[a,b]$. This concept is also valid if $\mathbf{F}$ is an interval vector-valued function of an interval vector $\mathbf{X}$, and complex interval arithmetic can also be defined. These facts should enable us to computationally but rigorously check the stability of systems. (See Section 4 below.)

If available, the language Fortran-SC is a convenient way of programming interval arithmetic computations. This precompiler is available on IBM mainframe equipment, and requires the ACRITH subroutine package; see [3] or [27]. There is also Pascal-SC for personal computers; see [22]. For a discussion of other programming languages and packages for interval arithmetic, see [13].

3 Nonlinear systems and global optimization

We briefly outline the principles which interval arithmetic uses to solve the problem

(3.1) Find, with certainty, approximations to all solutions of the nonlinear system

$$F(X) = (f_1(x_1, x_2, \ldots, x_n), \ldots, f_n(x_1, x_2, \ldots, x_n)) = 0,$$

where bounds l_i and u_i are known such that

$$l_i \le x_i \le u_i \text{ for } 1 \le i \le n.$$

and to solve the related problem

(3.2) Find, with certainty, the global minimum of the nonlinear objective function

$$\Phi(X)$$

subject to the constraints

$$l_i \le x_i \le u_i \text{ for } 1 \le i \le n.$$

We discuss generalized bisection in conjunction with interval Newton methods for solving (3.1) and (3.2). See chapters 19 and 20 of [1] chapters 5 and 6 of [16]. Early papers on the technique include [6], [20], and [14]; other papers are (but are not limited to) [8], [9], [15], [7], and [21].

We denote the box in n-space described by

$$\{X = (x_1, x_2, \ldots, x_n) \mid l_i < x_i < u_i \text{ for } 1 \le i \le n\}$$

by $\mathbf{X}$, and we generally use capital boldface letters for vectors whose entries are intervals. In interval Newton methods, we find a box $\overline{\mathbf{X}}_k$ which contains all solutions of the interval linear system

$$\mathbf{F}'(\mathbf{X}_k)(\overline{\mathbf{X}}_k - X_k) = -F(X_k), \tag{3.3}$$

where $\mathbf{F}'(\mathbf{X}_k)$ is an elementwise interval extension of the Jacobian matrix. We then define the next iterate $\mathbf{X}_{k+1}$ by

$$\mathbf{X}_{k+1} = \mathbf{X}_k \cap \overline{\mathbf{X}}_k. \tag{3.4}$$

The scheme based on solving (3.3) and performing (3.4) is termed an *interval Newton method.*

If each row of $\mathbf{F}'$ contains all possible vector values that that row of the scalar Jacobian matrix takes on as X ranges over all vectors in $\mathbf{X}_k$, then it follows from the mean value theorem that all solutions of (3.1) in $\mathbf{X}_k$ must be in $\mathbf{X}_{k+1}$. If the coordinate intervals of $\mathbf{X}_{k+1}$ are smaller than those of $\mathbf{X}_k$, then we may iterate (3.3) and (3.4) until we obtain an interval vector the widths of whose components are smaller than a specified tolerance.

If the coordinate intervals of $\mathbf{X}_{k+1}$ are not smaller than those of $\mathbf{X}_k$, then we may bisect one of these intervals to form two new boxes; we then continue the iteration with one of these boxes, and push the other one on a stack for later consideration. After completion of the current box, we pop a box from the stack, and apply (3.3) and (3.4) to it; we thus continue until the stack is exhausted. As is explained in [15], [8], and elsewhere, such a composite generalized bisection algorithm will reliably compute all solutions to (3.1) to within a specified tolerance.

Neumaier shows in [19] for many methods of solving (3.3),

(3.5) if $\overline{\mathbf{X}}_k \subset \mathbf{X}_k$, then the system of equations in (3.1) has a unique solution in $\mathbf{X}_k$. Conversely, if $\overline{\mathbf{X}}_k \cap \mathbf{X}_k = \emptyset$ then there are no solutions of the system in (3.1) in $\mathbf{X}_k$.

In many such cases where $\overline{\mathbf{X}}_k \subset \mathbf{X}_k$, we may also conclude that Newton's method starting from any point in $\mathbf{X}_k$ will converge to that solution. Furthermore, if directed roundings as mentioned in Section 2 are used, such conclusions are mathematically rigorous.

Iteration with formulas (3.3) and (3.4) should exhibit the quadratic local convergence properties of Newton's method, but repeated bisections are to be avoided if possible. We thus want an good interval extension to the Jacobian matrix, and we are interested in arranging the computations so that $\overline{\mathbf{X}}_k$ has coordinate intervals which are as narrow as possible.

The linear interval equation (3.3) may be solved with an interval version of the Gauss-Seidel method. In that and other methods for solving (3.3), we customarily first multiply both sides of (3.3) by a nonsingular preconditioner matrix Y_k to cause or accelerate local convergence;we thus obtain a related system

$$\mathbf{G}(\overline{\mathbf{X}}_k - X_k) = -h, \tag{3.6}$$

where $Y_k F(X_k) = h$ and $Y_k \mathbf{F}'(\mathbf{X}_k) = \mathbf{G}$.

Upon request, the author will send a detailed example of application of the interval Gauss-Seidel method to solve (3.6), which illustrates the principle (3.5).

The global nonlinear optimization problem (3.2) can be approached by solving (3.1), where $F = \nabla \mathit{\Phi}$. However, we may use the objective function directly to increase the algorithm's efficiency. In particular if $\mathbf{X}$ and $\mathbf{Y}$ are interval vectors in the stack described below (3.4), and Φ is an interval extension to $\mathit{\Phi}$, then, if $\Phi(\mathbf{Y}) > \Phi(\mathbf{X})$, we may discard $\mathbf{Y}$ from the stack. (We say that that $\mathbf{p} > \mathbf{q}$ if every element of $\mathbf{p}$ is greater than every element of $\mathbf{q}$.)

Walster, Hansen, and Sengupta report performance results on their interval global optimization algorithm in [26]; see [11] for other references.

4 Testing for nonnegative roots, etc.

Suppose $\tilde{g}: \quad \mathcal{C} \rightarrow \mathcal{C}$, where $\mathcal{C}$ is the complex plane, and suppose we wish to

(4.1) determine whether $\tilde{g}$ has any roots which occur in the right half plane $\{w \in \mathcal{C} \mid \Re(w) > 0\}$, where $\Re(w)$ is the real part of w.

We propose two approaches; details will appear elsewhere. In both, we define interval arithmetic on complex numbers by identifying a complex interval $\tilde{\mathbf{c}}$ with an ordered pair $(\mathbf{a}, \mathbf{b})$ of real intervals; we then define complex interval arithmetic by extending the standard complex operations in the natural way.

In the first approach to (4.1), we take a conformal map φ from the unit circle to $\mathcal{C}$. (see a reference on complex variables or [25], p. 185.), and we define

$$\tilde{f}(z) = \tilde{g}\big(\varphi(z)\big).$$

We make an interval extension $\tilde{\mathbf{f}}$ of $\tilde{f}$, then extend $\tilde{f}$ to complex intervals $\tilde{\mathbf{z}} \in \big([-1,1],[-1,1]\big)$ by defining

$$\tilde{\mathbf{f}}(\tilde{\mathbf{z}}) = \tilde{\mathbf{f}}(\tilde{\mathbf{u}}), \quad \text{where}$$

(4.2) $\tilde{\mathbf{u}}$ is the largest interval contained in both $\tilde{\mathbf{z}}$ and $\big([-1,1],[-1,1]\big)$,

and by using extended interval arithmetic for infinite intervals. (See [16], pp. 66-68.) We then use the techniques described in Section 3 to determine a root of $\tilde{f}$ (or lack thereof).

The second approach to (4.1), is valid if $\tilde{g}$ is a polynomial. (The basic idea comes from a conversation the author had with Alexander Morgan.) In that case, we may reduce the search over the right half plane to a search over $\big([-1,1],[-1,1]\big)$ by homogenizing the equation, then working in complex projective space. See [25] for details of the homogenization process.

A possible second application of interval mathematics may be in checking the Nyquist criterion for stability of a system, as described in Section 5.3 of [2]. The Nyquist criterion can be checked by computing the topological degree of the map with respect to the Nyquist locus, while interval arithmetic allows us to do this rigorously on a computer. We will give details elsewhere.

5 Are these techniques really practical?

The techniques described above may be easily implemented with Fortran-SC (in [27]) or Pascal-SC (in [22]). Also, in [12] we describe and make available well-documented, self-contained, portable Fortran-77 software which will solve polynomial systems of equations without programming, and which can solve more general systems if the user is willing to program interval extensions of the function and Jacobian matrix.

Whether the techniques so implemented will be practical for a given instance of (3.1) or (3.2) is a more difficult question to answer. Computational evidence appears in [26] and in [9]. We discuss interval methods *vis á vis* alternate techniques in [1], but further work needs to be done.

If a correctly programmed algorithm based on the methods of Section 3 completes, then it cannot give incorrect conclusions. However, for certain F, the algorithms may take so much computation time that they are impractical. For which F this is so depends on interplays between the number of variables, the nonlinearities of the components of F, the condition number of the Jacobian matrix near the roots, and how involved the interval extensions are. We are fairly confident that the methods are practical for single polynomials of a reasonable degree. However, there are systems of cubics and quartics in six variables which seem to be very difficult. (We have at present more personal experience with (3.1) than (3.2); (3.2) may be a somewhat easier problem in general.)

We suggested above that interval methods might be applicable to discretizations of optimal control problems. Such discretizations would be large, sparse nonlinear optimization problems. Largeness per se is not bad; Schwandt solves nonlinear elliptic problems with the technique in [24]. However, Schwandt limited himself to a somewhat special class of problems, which exhibit an interval generalization of diagonal dominance. This obviates the need for preconditioner matrices Y_k as described below (3.5).

In practical problems, the function F often requires large programs and extensive computational effort to evaluate. (For example, in a shooting method, the function values are the result of integrating a system of ordinary differential equations.) With state-of-the-art tools, such programs can use interval arithmetic. In such cases, the interval values may not be good in the sense that they are only crude bounds on the actual range of the function. More experience with this type of problem is desirable.

It is clear that we may make improvements to widen the range of applicability to ill-conditioned problems, large problems, and problems for which the degree of nonlinearity varies across the components. A promising approach is to design better preconditioner matrices Y_k. An example of this appears in [10].

References

[1] G. ALEFELD and J. HERZBERGER, *Introduction to Interval Computations*, Academic Press, New York, etc., 1983.

[2] BARNETT, S., *Introduction to Mathematical Control Theory*, Clarendon Press, Oxford, 1975.

[3] J. H. BLEHER, S. M. RUMP, U. KULISCH, M. METZGER, and W. WALTER, "Fortran-SC – A Study of a Fortran Extension for Engineering Scientific Computations with Access to ACRITH", *Computing*, v. 39, 1987, pp. 93–110.

[4] J. GARLOFF, "Interval Mathematics: A Bibliography", preprint, Institut für Angewandte Mathematik der Universität Freiburg, *Freiburger Intervall-Berichte*, v. 85, 1985, pp. 1–222.

[5] J. GARLOFF, "Bibliography on Interval Mathematics, Continuation" preprint, Institut für Angewandte Mathematik der Universität Freiburg, *Freiburger Intervall-Berichte*, v. 87, 1987, pp. 1–50.

[6] E. R. HANSEN, "On Solving Systems of Equations Using Interval Arithmetic", *Math. Comp.*, v. 22, 1968, pp. 374–384.

[7] E. R. HANSEN, "An Overview of Global Optimization Using Interval Analysis", in *Reliability in Computing*, R. E. Moore, Ed., Academic Press, New York, 1988.

[8] R. B. KEARFOTT, "Abstract Generalized Bisection and a Cost Bound", *Math. Comp.*, v. 49, 1987, pp. 187–202.

[9] R. B. KEARFOTT, "Some Tests of Generalized Bisection", *ACM Trans. Math. Software*, v. 13, 1987, pp. 197–220.

[10] R. B. KEARFOTT, "Preconditioners for the Interval Gauss-Seidel Method", submitted to *SIAM J. Numer. Anal.*, 1988.

[11] R. B. KEARFOTT, “Interval Arithmetic Techniques in the Computational Solution of Nonlinear Systems of Equations: Introduction, Examples, and Comparisons”, to appear in the proceedings of the 1988 AMS-SIAM Summer Seminar in Applied Mathematics, Colorado State University, July 18-29, 1988.

[12] R. B. KEARFOTT, and M. NOVOA, “A Program for Generalized Bisection”, submitted to *ACM Trans. Math. Software*, 1988.

[13] R. B. KEARFOTT, “Interval Arithmetic Methods for Monlinear Systems and Nonlinear Optimization: An Outline and Status”, to appear in *The Impact of Recent Computer Advances on Operations Research*, Elsevier, New York, 1989.

[14] R. E. MOORE, “A Test for Existence of Solutions to Nonlinear Systems”, *SIAM J. Numer. Anal.*, v. 14, 1977, pp. 611–615.

[15] R. E. MOORE, and S. T. JONES, “ Safe Starting Regions for Iterative Methods”, *SIAM J. Numer. Anal.*, v. 14, 1977, pp. 1051–1065.

[16] R. E. MOORE, *Methods and Applications of Interval Analysis*, SIAM, Philadelphia, 1979.

[17] R. E. MOORE, ed., *Reliability in Computing*, Academic Press, New York, etc., 1988.

[18] A. P. MORGAN, *Solving Polynomial Systems Using Continuation for Engineering and Scientific Problems*, Prentice-Hall, Englewood Cliffs, NJ, 1987.

[19] A. NEUMAIER, “Interval Iteration for Zeros of Systems of Equations”, *BIT*, v. 25, 1985, pp. 256-273.

[20] K. NICKEL, “On the Newton Method in Interval Analysis”, Technical Report 1136, Mathematics Research Center, University of Wisconsin, Madison, 1971.

[21] K. NICKEL, “Optimization Using Interval Mathematics”, preprint, Institut für Angewandte Mathematik der Universität Freiburg, *Freiburger Intervall-Berichte*, v. 86, 1986, pp. 55–83.

[22] L. B. RALL, “An Introduction to the Scientific Computing Language Pascal-SC”, *Comput. Math. Appl.*, v. 14, 1987, pp. 53–69.

[23] H. RATSCHEK and J. G. ROKNE, *Computer Methods for the Range of Functions*, Horwood, Chichester, England, 1984.

[24] H. SCHWANDT, "An Interval Arithmetic Approach for the Construction of an Almost Globally Convergent Method for the Solution of the Nonlinear Poisson Equation", *SIAM J. Sci. Statist. Comput.*, v. 5,1984, pp. 427–452.

[25] F. STENGER, "Numerical Methods Based on Whittaker Cardinal, or Sinc Functions", *SIAM Rev.*, v. 23, 1981, pp. 165–223.

[26] G. W. WALSTER, E. R. HANSEN, and S. SENGUPTA, "Test Results for a Global Optimization Algorithm", in *Numerical Optimization 1984*, (Boulder, Colo., June 12–14), SIAM, Philadelphia, 1985, pp. 272–287.

[27] W. WALTER and M. METZGER, "Fortran-SC, A Fortran Extension for Engineering/Scientific Computation with Access to ACRITH", in *Reliability in Computing*, R. E. Moore, ed., Academic Press, New York, etc., 1988.