Reliable Computing 1 (3) (1995), pp. 215–223

Application of interval Newton's method to chemical engineering problems

Gopalan V. Balaji and J. D. Seader

The interval Newton's method in conjunction with generalized bisection, as implemented in the public domain software program INTBIS by Kearfott and Novoa, is a method of solving single and simultaneous nonlinear equations. In this paper, this method is used to solve 15 test problems from different chemical engineering application areas, and an ill-scaled transcendental equation. The interval method as implemented in INTBIS is capable of finding all the real roots of an equation within a specified domain. The complex roots were found by representing every complex variable $z = x + iy$ by two real-valued variables x and y. For polynomial equations, the computer time for INTBIS is compared to that of the parallel-path continuation method of Morgan as implemented in the public domain software program CONSOL8.

Unlike the parallel-path continuation method, that can find all real and complex roots without specifying an initial domain, INTBIS requires the apriori specification of an initial domain. The effect of the size of the initial domain on the computer time for INTBIS was studied for a set of multi-variable polynomial equations.

Приложение интервального метода Ньютона к задачам химической инженерии

Г. В. Баладжи, Дж. Д. Сидер

Интервальный метод Ньютона в совокупности с обобщенным методом деления пополам, реализованные в свободно доступном пакете программ INTBIS Кирфотта и Новоа, предназначены для решения нелинейных уравнений и их систем. В настоящей работе этот метод применяется для решения 15 тестовых задач из различных областей химической инженерии, а также плохо масштабированного транцендентального уравнения. Интервальный метод, реализованный в INTBIS, способен находить все вещественные корни уравнения в заданной области. Комплексные корни искались путем представления каждой комплексной переменной $z = x + iy$ в виде двух вещественных переменных x и y. Для полиномиальных уравнений время счета в системе INTBIS сравнивалось с временем счета по методу продолжения параллельных траекторий Моргана, реализованному в свободно доступном пакете программ CONSOL8.

В отличие от метода продолжения параллельных траекторий, который может находить все вещественные и комплексные корни без указания начальной области, INTBIS требует априорного задания начальной области. Влияние размера этой начальной области на время счета для INTBIS было изучено на наборе полиномиальных уравнений многих переменных.

1. Introduction

1.1. In engineering applications, it is often necessary to find the roots of an equation or a system of equations

Many engineering problems can be reduced to finding the solutions of a nonlinear equation (or system of equations) $f(x) = 0$, usually with polynomial f. Many algorithms have been developed for solving these equations.

A polynomial equation of n-th order has (in general) n complex solutions (n roots of the polynomial f). Only some of them are physically meaningful; usually, only real solutions (but not all real solutions) are meaningful.

A typical example is a state equation. State equations usually originate from the condition that some objective function $V(x)$ (e.g., potential energy) attains local minimum. To describe local minima, we can write an equation $V'(x) = 0$. This equation, however, describes not only local minima, but local maxima as well, and local maxima correspond to highly unstable (and thus, physically impossible) states. For example, a cubic equation describes the molar volume of a fluid. This equation has three solutions, but only two of these solutions have a physical meaning [10].

In *chemical engineering*, root finding is a very important and often used step during process design and flowsheeting. Process design computations involve mathematical modeling of a specific operating unit in a chemical plant. Flowsheeting involves mathematical modeling of the entire chemical plant with all the operating units in it interacting with each other in a very complex manner. Commercial software that simulates process flowsheets employ root-finding techniques during each iteration while solving nonlinear design equations.

1.2. To guarantee that all roots have been found, we need to find *all* the roots

Suppose that we are using a numerical method to find a solution to a given equation $f(x) = 0$. We are interested only in physically meaningful solutions, e.g., real solutions that belong to a given interval. There are two possible outcomes:

- the method can return a value (or several values) that are supposed to be the desired solutions; or
- the method can fail, indicating that probably the given system has no solutions within given bounds.

We are saying "supposes to" and "probably", because we are usually using numerical methods that do not give guaranteed results. What *can* we guarantee in these cases?

- If the method resulted in a value x, then we can easily check whether this x is indeed a solution: first, we check that this x belongs to a given interval, and second, we substitute x into f and check that $f(x)$ is indeed equal to 0 (or rather, taking into consideration the fact that computers have finite precision, that $f(x)$ is sufficiently close to 0). If the supposed solution x survives this test, we thus get a value that is *guaranteed* to be a solution to our problem.
- The situation in which the method failed to find the solution is more difficult to check. We would like to know whether there are no solutions at all, or whether there are solutions, but the method failed to find them.

For general (transcendental) equations, this is difficult to find out. However, for *polynomial* equations, there is a way: namely, we can use the fact that a polynomial equation of n-th order has exactly n complex roots (if we count multiple and higher order roots corresponding

number of times). So, if we find all n complex roots, and we check that none of them belongs to the desired interval, then we can be sure that on the given interval, there are no solutions to the given equation.

Similarly, to guarantee that we have found all physically meaningful solutions to a given equation, it is desirable to find *all* solutions, and check that other solutions are not in the desired area.

In view of that, it is important to be able to find *all* solutions (real and complex) of a given equation, even if we know that only real solutions have a physical meaning.

1.3. The existing methods of finding the solutions of nonlinear equations

The existing methods of finding the roots of nonlinear equations can be crudely divided into two groups:

- *Local* methods, such as Newton's method. These methods start in the vicinity of a root, and find at best one root. These method usually converge very fast, but they are not 100% reliable in the sense that they are not guaranteed to find even one root, not to say all of them.
- *Global* methods, that try to find *all* the roots without requiring any initial guesses. An example is a *continuation method* [1, 7]. These methods usually require an order-of-magnitude more computation time than local methods, but with the speed of today's computers, this time is becoming less and less a factor in choosing a method.

For engineering problems, we usually have a priori bounds on the roots. Therefore, it is desirable to use fast local methods. The only reason why at present these methods are not always used is that, as we have mentioned, the majority of existing local methods are unreliable. So, the ideal solution would be to find *reliable* local methods, i.e., fast local methods that result in guaranteed bounds for the solutions. Such methods are provided by interval computations.

1.4. Round-off errors and interval computations

Interval computations originated from the following problem. Computers use finite (approximate) representations of real numbers, and therefore, all computer operations with real numbers are approximate. The resulting round-off errors build up, leading to inaccurate results.

In particular, in chemical engineering design, an iterative process is used that employs root-finding techniques during each iteration. The number of iterations may be large, even for nominal-sized problems. On each iteration, an additional round-off error is added to the result. At the end, these small round-off errors can add up to a reasonable-size error.

When the initial data are precisely known, and the numerical method is exact, then round-off is the only source of error. In this case, if we increase the length of the machine word (and thus, the precision of computer operations), the width of the resulting interval decreases (and tends to 0 as the length of the machine word goes to ∞).

In real computers, we must take round-off and propagation error into consideration. One way to do that is to use interval arithmetic. In interval arithmetic, the basic object is not a

number, but an interval $[a, b]$. A number that is precisely known is represented by a degenerate interval $[a, a]$. A number whose value is not precisely known is represented by an interval of possible values [6]. The result of applying interval computations techniques is not an *approximate* value $\tilde{x}$ of the desired parameter x, it is an *interval* $\mathbf{x}$ that is guaranteed to contain the (unknown) value x. This interval takes into consideration the finite precision of the computers. Due to this fact, the number of applications of interval computations to solving equations is growing.

In particular, for engineering problems, in which we know the approximate locations of the roots, we may want to apply interval local methods to compute the resulting guaranteed estimates fast. In this paper, we use the (multi-dimensional) interval Newton's method for solving a system of nonlinear equations. This method, in conjunction with the generalized bisection method, is implemented in the public domain software INTBIS [5].

1.5. Results

In chemical engineering, as we have already mentioned, round-off errors can be large for process design and flowsheeting. Therefore, interval methods seem an attractive choice for these problems. In this paper, we will describe the results of applying interval methods to checmical engineering problems.

The INTBIS package itself only finds *real* roots of *polynomial* equations. The availability of transcendental functions in the interval arithmetic library INTLIB [4] enabled us to modify INTBIS so that it can solve *transcendental* equations as well. To find *complex* roots z, we represent each complex variable $z = x + iy$ by a pair of real variables x and y, and apply INTBIS to the resulting system.

We applied the resulting techniques to 14 chemical engineering problems described in [9] (these problems lead to nonlinear equations typical for chemical engineering). INTBIS was also applied to a chemical engineering problem in which the function f in the equation $f(x) = 0$ is *rational* (and not polynomial) [1], and (to test the method) to the equation $f(x) = 0$ with a *transcendental* function f [11]. In order to test the method on a system of linear and nonlinear equations, we tried it on a two-stage reactor problem [8].

2. Method

2.1. INTBIS

INTBIS [5] is based on the interval Newton/generalized bisection algorithm that combines a geometric bisection method with the interval Newton's method (this algorithm is explained in [3]).

In order to solve the system of (nonlinear) equations $F(X) = 0$ on a given box $\mathbf{X}$, we subdivide this box into smaller boxes. To each box, the interval Newton's method is applied. This is an iterative method, in which the box $\mathbf{X}^{k+1}$ on the next iteration is obtained from the box $\mathbf{X}^k$ computed on the previous iteration as $\mathbf{X}^{k+1} = \mathbf{X}^k \cap \tilde{\mathbf{X}}^{k+1}$, where $\tilde{\mathbf{X}}^{k+1}$ is computed by solving the following system of linear interval equations:

$$\mathbf{F}'(\mathbf{X}^k)(\tilde{\mathbf{X}}^{k+1} - X^k) = -\mathbf{F}(\mathbf{X}^k)$$

$\mathbf{F}(\mathbf{X}^k)$ and $\mathbf{F}'(\mathbf{X}^k)$ are the interval extensions of F and the Jacobian F' evaluated at $\mathbf{X}^k$, and X^k is a midpoint of the box $\mathbf{X}^k$. To solve this system, the interval Gauss-Seidel technique is used. If for a certain subbox, this method does not lead to a definite conclusion about whether this box contains a root or not, then this box is further bisected. Finally, we get boxes for each of which the interval Newton method either converges to a solution, or leads to $\mathbf{X}^k = \varphi$, meaning that this subbox has no solutions in it.

This approach combines the exhaustiveness of the bisection method with the speed of the Newton's method.

2.2. Handling complex roots: idea and example

Classical Newton's method can be easily generalized to complex numbers. However, the *interval* Newton method is derived from the mean-value theorem, which does not hold in the complex domain [2]. Therefore, we cannot directly generalize this method to finding complex solutions.

Instead, we apply the following natural idea: we replace each complex variable $z = x + iy$ by two real variables x and y. Correspondingly, instead of a single complex equation $F = 0$, we consider two real equation that represent the real and imaginary parts of the complex one. Let us illustrate this idea on equation 1 from Shacham [9]. This quartic equation describes the fraction of the nitrogen-hydrogen feed that gets converted to ammonia (this fraction is called *fractional conversion*). For 250 atm and 500°C, this equation takes the form

$$f(z) = z^4 - 7.79075z^3 + 14.7445z^2 + 2.511z - 1.674 = 0.$$

If we substitute $z = x+iy$ into this equation, expand $(x+iy)^k$, and separate real and imaginary terms, we get the following equation:

$$\begin{aligned}(x^4 - 6x^2y^2 + y^4 - 7.79075x^3 + 23.37225xy^2 + 14.7445x^2 - 14.7445y^2 + 2.511x - 1.674)\\ + (4x^3y - 4xy^3 - 23.37225x^2y + 7.79075y^3 + 29.489xy + 2.511y)i = 0.\end{aligned}$$

For a complex number to be equal to 0, both its real and imaginary parts must be equal to 0. So, we arrive at the following system of two equations:

$$\begin{aligned}x^4 - 6x^2y^2 + y^4 - 7.79075x^3 + 23.37225xy^2 + 14.7445x^2 - 14.7445y^2 + 2.511x - 1.674 &= 0,\\ 4x^3y - 4xy^3 - 23.37225x^2y + 7.79075y^3 + 29.489xy + 2.511y &= 0.\end{aligned}$$

For this system, INTBIS computed narrow intervals that contain all 4 known roots (these roots can obtained, e.g., from the known analytical solution of a quartic equation): $z_1 = 0.278$, $z_2 = -0.384$, $z_3 = 3.949 + 0.316i$, $z_4 = 3.949 - 0.316i$.

By definition, the fractional conversion is a number between 0 and 1. Therefore, only the root z_1 is physically meaningful.

3. Tests and their results

3.1. A list of test problems

Both INTBIS and CONSOL8 were applied to:

1) 14 out of 15 chemical engineering problems presented in Shacham [9].
2) A Continuous Stirred Tank Reactor (CSTR) problem due to Seader et al. [8].

3) An equation with a rational f described in Gritton [1]; and

4) A purely mathematical transcendental equation suggested by Watson [11].

The computations were done on SUN Sparc10 stations. Let us describe the results for each of these groups of problems.

3.2. Problems from Shacham

Problems 1, 2, 4, 5, 6, 11, 12, 13, and 15 of Shacham [9] were reduced to polynomial form. The most complicated problem was problem 2, in which Rachford-Rice equation for the isothermal-flash calculation of a 19-component system results in an equation $f(x) = 0$ with an 18th order polynomial. This system has 18 real roots in the interval $[-12, 7]$; only one of these roots belongs to the interval $[0, 1]$, and is, therefore, physically meaningful. Problems 3, 7 and 10 involved transcendental terms, while problems 8 and 9 involved fractional powers of the dependent variable. Problem number 14 of Shacham was not included because it can be converted to a linear equation.

For problems with polynomial f, apriori bounds for roots are known:

1A	Ammonia synthesis	$[-1, 4]$	$+[-1, 1]i$
1B	Ammonia synthesis	$[-3, 5]$	$+[0, 3]i$
4	Azeoteric point	$[0, 6]$	
5	Adiabatic flame temp	$[-9000, 5000]$	$+[-9000, 9000]i$
6	Beattie-Bridgeman EOS	$[0, 2]$	$+[-5, 5]i$
11	Chemical Equilibrium	$[0, 0.7]$	$+[-0.4, 0.4]i$
12	Viral EOS	$[-100, 300]$	$+[-100, 100]i$
13	Redlich-Kwong EOS	$[0, 0.1]$	$+[-0.1, 0.1]i$
15	Sinkage depth of a sphere	$[-1, 3]$	

These bounds were given to INTBIS as the initial boxes. In problems 4 and 15, it is known that all the roots are real, so imaginary parts were not used.

Both INTBIS and CONSOL8 found all real and complex roots for each problem, except for problem number 2, where INTBIS had some difficulty in solving the 18th order polynomial: many subboxes were left unresolved after many bisections.

The CPU times (in sec) for solving the polynomial equations using INTBIS and CONSOL8 are as follows:

Problem		INTBIS	CONSOL8
1A	Ammonia synthesis	2.200	0.217
1B	Ammonia synthesis	2.717	0.183
4	Azeotropic point	0.033	0.133
5	Adiabatic flame Temp	0.150	0.167
6	Beattie-Bridgeman EOS	1.267	0.200
11	Chemical Equilibrium	3.117	0.183
12	Viral EOS	1.050	0.133
13	Redlich-Kwong EOS	0.400	0.150
15	Sinkage depth of a sphere	0.033	0.167

3.3. Problems from Seader et al.

In Seader et al. [8], continuation methods are used to solve a system of nonlinear and linear equations resulting from the 1985 AIChE Student Contest Problem. The example describes an acid-catalyzed esterification reaction carried out continuously in two reactors. The resulting system of equations consists of two linear equations, four quadratic equations, and one cubic equation. According to Bezout's Theorem [7], such a system has (in general) 48 roots. For this particular system, only one root has a physical significance, all the others are either complex with non-zero imaginary parts, or negative.

On our Sparc10 workstation, CONSOL8 found all 48 roots in about 134 sec. INTBIS was used to determine the real positive root only. The following initial box for variables x_i was determined by apriori knowledge and supplied to INTBIS:

x_1	x_2	x_3	x_4	x_5	x_6	x_7
[70, 90]	[6, 8]	[50, 70]	[6, 8]	[0, 1]	[3, 4]	[0, 1]

It took INTBIS about 32 sec to find the desired root.

The apriori box spans approximately 20% on either side of the root. If we did not have this apriori information, we would end up with an apriori box spanning 40% to 50% on either side of the root. In this case, INTBIS CPU time increases to 100–150 sec.

3.4. A problem by Gritton

Gritton [1] used a global-fixed point homotopy method to calculate the chemical equilibrium in ammonia synthesis (above-described problem 1 from [9]). The original problem consists of solving the equation

$$\frac{8(4-x)^2x^2}{(6-3x)^2(2-x)} - 0.186 = 0.$$

In the previous approach, we reduced this problem to the polynomial equation. However, a modified INTBIS can be applied to the original equation as well.

This equation has a singularity at $x = 2$, when the denominator becomes 0. The initial (apriori) interval for x is $[-5, 2.5]$. INTBIS found both real roots $x = -0.384$ and $x = 0.278$, and also found the singularity at 2.

3.5. Testing on a transcendental equation

Watson [11] suggested the following transcendental equation: $\exp(-x^2) \cdot \sin(x) = 0$. The roots of this equation coincide with the roots of the equation $\sin(x) = 0$, but the exponential term introduces a large variation of the function in the interval $[-25, 25]$. INTBIS found all the roots in this interval. Outside this interval, the left-hand side cannot be computed even with double precision variables: it produces either an underflow error (for $x < -25$), or an overflow error (for $x > 25$).

3.6. Effect of size of initial region on the INTBIS CPU time

The effect of the size of initial region on the INTBIS CPU time was studied for the two-stage reaction problem of Seader et al. [8]. The results are as follows:

% on either side of root	Time (in sec)	`ftest` calls	function calls
1	1.68	43	90
2	3.35	123	191
5	8.43	412	477
10	20.73	1098	1188
20	39.62	2351	2302
30	58.00	3813	3404
40	90.67	6301	5198
50	152.5	11108	8729
75	213.0	15684	12018
100	465.2	33423	26558

The dependency of all three parameters (time, `ftest` calls, and function calls) on the width of the initial region was similar and approximately quadratic. Therefore, for this method, narrow initial bounds will lead to substantial saving of the CPU time.

4. Conclusions and recommendations

The facts that INTBIS searches for all the roots (exhaustively) and takes into consideration round-off errors makes INTBIS an attractive choice for application in chemical engineering design programs. INTBIS can be easily incorporated into programs through its main driver routine GENBIS. In spite of the fact that its interval arithmetic is software simulated and therefore, reasonably slow, for solving polynomial equations, the resulting CPU times are comparable with the CONSOL8 times. When only real solutions are desired, INTBIS requires less time than CONSOL8. With the possible availability of interval arithmetic processors, INTBIS running time will be substantially reduced, making it competitive for use in design and simulation software.

INTBIS has the added advantage of being applicable to transcendental equations.

For a system of several transcendental equations, we cannot recommend to use INTBIS in its present form, because programming the interval extensions of the functions and Jacobians in interval arithmetic is very tedious. For such systems, a modification is needed that would automatically create these extensions from the user-provided floating point expression.

Acknowledgments

We appreciate the assistance of Professor R. Baker Kearfott at every step of applying INTBIS to our equations. Mr. Jui-Jung Chen provided the technique for transforming the complex variables into two-dimensional real variables. Mr. Sanjay Sharma conducted some of the experiments with CPU times and initial bounds.

References

[1] Gritton, K. S. *Application of global fixed-point homotopy to single-nonlinear-equation chemical engineering problems*. Ph. D. Dissertation, University of Utah, 1991.

[2] Hansen, E. *A globally convergent interval method for computing and bounding real roots.* BIT **18** (1978), pp. 415–424.

[3] Kearfott, R. B. *Abstract generalized bisection and a cost bound.* Math. Comput. **49** (179) (1987), pp. 187–202.

[4] Kearfott, R. B., Dawande, M., Du, K., and Hu, C. *INTLIB: a portable Fortran-77 elementary function library.* Accepted for publication as an algorithm in the ACM Transactions on Mathematical Software.

[5] Kearfott, R. B. and Novoa III, M. *INTBIS, a portable interval Newton/bisection package.* ACM Trans. Math. Software **16** (2) (1990), pp. 152–157.

[6] Moore, R. E. *Methods and applications of interval analysis.* SIAM, Philadelphia, PA, 1979.

[7] Morgan, A. P. *Solving polynomial systems using continuation for engineering and scientific problems.* Prentice-Hall, Englewood Cliffs, NJ, 1987.

[8] Seader, J. D., Kuno, M., Lin, W.–J., Johnson, S. A., Unsworth, K., and Wiskin, J. S. *Mapped continuation methods for computing all solutions to general systems of nonlinear equations.* Computers Chem. Engng. **14** (1) (1990), pp. 71–85.

[9] Shacham, M. *An improved memory method for the solution of a nonlinear equation.* Chem. Eng. Sci. **44** (7) (1989), pp. 1495–1501.

[10] Smith, J. M. and Van Ness, H. C. *Introduction to chemical engineering thermodynamics.* Fourth Edition, McGraw-Hill Book Company, NY, 1987, pp. 80–84.

[11] Watson, L. Personal communication, 1988.

Received: March 1, 1994
Revised version: May 19, 1994

Department of Chemical & Fuels Engineering
University of Utah
Salt Lake City
UT 84112
USA