Towards the future of interval computations

Editors' Introduction to the Student Issue

The idea of the student issue. The future of Interval Computations is with the *student researchers* entering the field. Their vision, their insights, their approaches will shape the future research. To highlight their contributions, the Editorial Board of the International Journal *Interval Computations* (since 1995, *Reliable Computing*) decided to make a special issue with high-quality papers written by students (either alone, or in co-authorship with their professors). The call for papers was issued at the end of 1993, and finally, after the thorough iterative process of refereeing and revision (these papers have undergone the same refereeing as all the other papers), we are happy to finally present this issue to the readers.

The idea of the student issue turned out to be very successful. The total volume of accepted papers far exceeded a typical issue size, so some papers are moved to other issues [2, 7, 8], and several submitted papers are still under consideration. In view of this success, we suggest to make such student issues regular.

Let us briefly describe the contents of this issue.

Solving systems of equations: one of the main problem of numerical mathematics, and how interval methods help. One of the main problems of numerical mathematics is to solve a system of equations $f_i(x_1, \ldots, x_n) = 0$, $1 \le i \le n$.[1]

For non-linear equations, it is rarely possible to compute the exact solutions in finitely many computation steps, so traditional numerical methods (e.g., bisection or Newton's method) only give us *approximate* solutions. These methods rarely provide us with an *accuracy* of these approximate solutions (i.e., with the guaranteed bounds for the difference between the approximate solution and the desired exact ones). To get *guaranteed* bounds (i.e., to get *intervals* that contain the desired values) special (interval) methods are needed. In this issue, the following methods are described and used:

[1] Another large group of problems are optimization problems $f(x_1, \ldots, x_n) \to \max$, but, e.g., for smooth objective functions, these problems reduce to solving a system of equations $\partial \tilde{f}/\partial x_i = 0$, where:

- for unconditional optimization, $\tilde{f} = f$, and
- for conditional optimization under conditions $g_i = 0$, $\tilde{f} = f + \sum \lambda_i g_i$

- For *generic* systems, *interval Newton's/bisection method* is described and used in [5] and [6]. The resulting algorithms are applied to *design* and *analysis* in *chemical engineering* (e.g., what is the output of ammonia synthesis problem for given conditions).

- For *systems of polynomial equations*, a new interval method is proposed in [7]. This method reduces the original system of n equations with n variables to a *triangular* system, in which, crudely speaking:

  - the first equation has only one unknown, and
  - as soon as we have solved all equations $1, \ldots, i$, the $(i+1)$-st equation also has only one unknown.

  This method reduces the solution of a *system* of n equations with n unknowns to solving n equations with *one* unknown (and equations with one unknown are easier to solve than the system).

- When a system takes the form $f(z) = 0$ for a *complex-valued* (analytic) function of a *complex* variable $z = x_1 + x_2 \cdot i$, then we have two options:

  - We can apply interval Newton method [1]. In [1], the resulting algorithms are applied to *chemical engineering*.
  - We can also use the *argument principle*. This principle expresses the number of roots in an area via the integral over its boundary curve. By applying the resulting roots-counting algorithms to the subboxes obtained by appropriate bisection of the original box, we can find a sufficiently small box that contains the desired root z (if the equation $f(z) = 0$ has several roots $z_1, \ldots, z_k$, we find k small boxes that contain these roots).

  In this issue:

  - In [4], an interval algorithm is described that estimates the number of roots in a given area. This algorithm can be used to find these roots, or simply to check whether roots exist in a given area; the second possibility is used in *automatic control* to check if a given controlled system is stable.
  - In [13], an interval algorithm is described that actually finds all the roots.

Initial data can also be uncertain. In addition to inaccuracy of numerical methods, we may often have inaccurate initial data to begin with: this data comes from measurements or expert estimates, and measurements are never 100% accurate. As a result, we, e.g., have equations $\mathbf{f}_i(x_1, \ldots, x_n) = 0$ whose coefficients c_k are not precisely known: we only know the intervals $\mathbf{c}_k$ that contain the actual values of these coeffficients. In this case, we are interested in the intervals of possible values of solutions x_i.

- This problem becomes computationally non-trivial even if we have an *explicit* expression of x_i in terms of these coefficients: actually, even for some quadratic expressions $x_i = P_2(c_1, c_2, \ldots)$, it is impossible to compute the exact intervals for x_i in reasonable time. In [8], an efficient algorithm is proposed for *fractionally linear* expressions. This algorithm is applied to *intelligent control*.

- The simplest possible systems of equations are *linear* systems $\sum a_{ij}x_j = b_i$. If we only know intervals $\mathbf{a}_{ij}$ and $\mathbf{b}_j$ for a_{ij} and b_j, then in general case, the problem of finding the exact bounds for the interval of possible values of x_i is computationally intractable (for details, see [15]). Many real-life systems have specific features that make them easier to solve. Two such examples are given in the issue:
  - In many real-life cases, we have a *sparse* system (for which the majority of coefficients a_{ij} are 0). For sparse systems, an efficient algorithm is proposed in [5]. This algorithm is applied to *chemical kinetics* and *chemical engineering*.
  - In many real-life problems, measurement is so accurate that the difference $\Delta c_k = \tilde{c}_k - c_k$ between the measured $\tilde{c}_k$ and the actual c_k values of the coefficients is small. Therefore, we can effectively neglect terms that are quadratic in terms of this difference Δc_k. In this case, after we have found the solution $\tilde{x}_i$ of the system with the coefficients $\tilde{c}_k$, we get a *linearized* system to find the difference $x_i - \tilde{x}_i$, and we can use one step on Newton's method to solve this system. This method is described in [2]; as a case study, it is applied to *pavement engineering*.
- For *polynomial* equations with interval coefficients, an algorithm is proposed in [7]. This algorithm is based on reducing the original system of equations to the triangular system.

Interpolation: interval approach. In the above text, we assumed that we already know the dependencies f_i used in the equations. In many real-life situations, however, we do not know the exact dependencies; we only know the values $y^{(k)} = f_i(x_1^{(k)}, \ldots, x_n^{(k)})$ for some values $x^{(k)} = (x_1^{(k)}, \ldots, x_n^{(k)})$. To get the values $f_i(x_1, \ldots, x_n)$ for all $x = (x_1, \ldots, x_n)$, we must apply *interpolation*[2]. There are infinitely many ways to interpolate a function. For interval computations, it is natural to choose an interpolation that leads to the narrowest possible intervals of uncertainty in the result. This idea is described and formalized in three papers:

- In [9] and [12], a general case is considered, in which all functions f_i are apriori possible.
  - In [12], the results are applied to choosing knowledge representation techniques and operations with degrees of belief in *expert systems* and *intelligent control*.
  - In [9], algorithms are presented that check whether the interpolated dependency can be monotonic, and if not, what local extrema it can have. These algorithms are applicable to problems from *radioastronomy*, *spectroscopy*, *particle physics*, etc.
- In [3], a case is considered when we do have an apriori information about f_i: namely, we know that f_i must belong to one of the known finite-parametric classes of functions (e.g., it is either exponential, or polynomial). In this case, we choose a class that leads to the narrowest intervals. The resulting algorithms are applied to the analysis of *psychological data*.

In the above two papers, a *passive* interpolation problem is considered, in which we already have the data $(x^{(k)}, y^{(k)})$, and we want to find the best interpolating algorithm. In real life, we often have an *active* interpolation situation, in which we can choose for what points $x^{(k)}$ to measure the values $y^{(k)}$ of f_i. In this case, we can choose these points $x^{(k)}$ in such a way that the interval uncertainty of the resulting interpolation is the smallest possible. The

[2] or *extrapolation*, if the desired value x is outside the area formed by $x^{(k)}$.

problem of choosing interpolation points that are the best (i.e., that lead to narrowest intervals) is formulated and solved in [10], on the example of *coordinate-measuring machines* in industrial engineering.

Representing uncertainty in multi-dimensional case. In the above-cited papers, we assumed that we know the interval of possible values $\mathbf{c}_k$ of each coefficient c_k. In this case, possible values of the coefficient vector $c = (c_1, c_2, \ldots)$ form a rectangular *box* $\mathbf{c}_1 \times \mathbf{c}_2 \times \cdots$ In real life, some sources of error may be common to measurements to measuring several coefficients. As a result, not all combinations $c = (c_1, c_2, \ldots)$ are possible, and the actual set of possible values of c can be different from a box (e.g., it can be a parallelepiped, or an ellipsoid, etc).

The shape of this set may be too complicated for computational processing, so it can be approximated by sets of *simpler* shape.

In [11], it is shown that if we restrict the complexity of algorithms that process these sets, then *parallelepipeds* are the only possible choice. This result is applied to knowledge representation in *knowledge-based systems*.

Interval computations often require large computation time. Traditional numerical methods, that compute only the estimate for the desired value(s), often require many operations, and, therefore, take a lot of computation time. Interval analogues of these methods, in addition to the estimates themselves, compute accuracies of these estimates (i.e., intervals that contain the true values of the desired quantities). Because of the necessary extra computational steps, interval computations take even longer computation time.

Therefore, in many cases, it is important to speed up interval computations.

How to speed up interval computations? One way to speed up interval computations is to come up with a *faster algorithm*. This is, however, not always possible, because in general, the problems of interval computations are computationally intractable (NP-hard) (for exact definitions and formulations, see, [15]).

So, if we have an algorithm that cannot be made faster (or at least we do not know how to make it faster), then the only way to make it run faster is to *change the hardware* (i.e., the computer on which this algorithm runs). There are two possible ways to do that:

- First, we can use the *existing* faster *hardware*. A natural way to speed up computations is have several processors working *in parallel*. In this issue, parallelization is proposed:
  - in [5, 6], to solve systems of non-linear equations (*parallel computer*: CRAY);
  - in [5], to solve sparse systems of linear equations (*parallel computer*: CRAY);
  - in [13], to find roots of complex functions (*parallel computer*: a network of interconnected workstations).
- Second, we can design *new hardware*, new computer architecture that is specifically tailored for interval computations. Such a design is proposed in [14].

The fact that interval computations often require large computation time has a bright side. A typical proof that some problem P is very computationally complicated (i.e., that it requires a large computation time to solve) is by showing that some other problem P' (that is already known to be computationally complicated) can be reduced to solving particular cases of the problem P. Such proof uses a *negative* side of the reduction: since P' is difficult to solve, P is also difficult to solve.

It turns out that this reduction also has a *bright* side: if we have a heuristic that solves many instances of the problem P, then we may hope that by:

- reducing P' to P, and
- applying this heuristic to a resulting particular case of P,

we will also be able to solve important instances of the difficult-to-solve problem P'.

In [15], several known heuristics of interval computations are used in this manner to generate successful heuristics for the so-called *propositional satisfiability* problem that is known to be difficult-to-solve.[3]

Thanks. We would like to thank the authors for their excellent job, and the anonymous referees for their thorough and unrewarding job of reviewing the papers. We want to thank Slava Nesterov, who initially proposed the idea of the issue, who encouraged and supported us all this time, and who even handled refereeing of several papers from this issue. He can truly be called the third co-editor of this student issue. Finally, we want to thank the University of Texas at El Paso and NASA Grant No. 9–757 for financial support that helped make this issue possible.

References

[1] Balaji, G. V. and Seader, J. D. *Application of interval-Newton method to chemical engineering problems*. Reliable Computing **1** (3) (1995).

[2] Ferregut, C., Nazarian, S., Vennalganti, K., Chang, C. C., and Kreinovich, V. *Fast error estimates for indirect measurements: applications to pavement engineering*. Reliable Computing **2** (1996), to appear.

[3] Friesen, B. H. and Kreinovich, V. *Ockham's razor in interval identification*. Reliable Computing **1** (3) (1995).

[4] Herlocker, J. and Ely, J. *An automatic and guaranteed determination of the number of roots of an analytic function interior to a simple closed curve in the complex plane*. Reliable Computing **1** (3) (1995).

[5] Hu, Ch., Frolov, A., Kearfott, R. B., and Yang, Q. *A general iterative sparse linear solver and its parallelization for interval newton methods*. Reliable Computing **1** (3) (1995).

[6] Hu, Ch., Sheldon, J., Kearfott, R. B., and Yang, Q. *Optimizing INTBIS on the CRAY Y–MP*. Reliable Computing **1** (3) (1995).

[7] Jäger, C. and Ratz, D. *A combined method for enclosing all solutions of nonlinear systems of polynomial equations*. Reliable Computing **1** (1) (1995).

[8] Lea, R., Kreinovich, V., and Trejo, R. *Optimal interval enclosures for fractionally-linear functions, and their application to intelligent control*. Reliable Computing **2** (1996), to appear.

[3] Crudely speaking, propositional satisfiability problem is as follows:

Given: a *Boolean expression* F, i.e., the result of applying "and", "or", and "not" to Boolean (true-false) variables $v_1, \ldots, v_m$.

Find: the values of the variables $v_1, \ldots, v_m$ that make this expression F true

[9] Lorkowski, J. and Kreinovich, V. *If we measure a number, we get an interval. What if we measure a function or an operator?* Reliable Computing **2** (1996), to appear.

[10] McLean, T. J. and Xu, D. H. *Study on sampling techniques with CMMs.* Reliable Computing **1** (3) (1995).

[11] Misane, D. and Kreinovich, V. *A new characterization of the set of all intervals, based on the necessity to check consistency easily.* Reliable Computing **1** (3) (1995).

[12] Nguyen, H. T., Kreinovich, V., Lea, R., and Tolbert, D. *Interpolation that leads to the narrowest intervals, and its application to expert systems and intelligent control.* Reliable Computing **1** (3) (1995).

[13] Schaefer, M. J. and Bubeck, T. *A parallel complex zero finder.* Reliable Computing **1** (3) (1995).

[14] Schulte, M. and Swartzlander, E. E. *A hardware design and software interface for variable-precision interval arithmetic.* Reliable Computing **1** (3) (1995).

[15] Traylor, B. and Kreinovich, V. *A bright side of NP-hardness of interval computations: interval heuristics applied to NP-problems.* Reliable Computing **1** (3) (1995).

V. KREINOVICH

G. MAYER