1

APPLICATIONS OF INTERVAL COMPUTATIONS: AN INTRODUCTION

R. Baker Kearfott* and Vladik Kreinovich**

**Department of Mathematics, University of Southwestern Louisiana, U.S.L. Box 4-1010, Lafayette, LA 70504-1010, USA, email: rbk@usl.edu*

***Department of Computer Science, University of Texas at El Paso, El Paso, TX 79968, email vladik@cs.utep.edu*

ABSTRACT

The main goal of this introduction is to make the book more accessible to readers who are not familiar with interval computations: to beginning graduate students, to researchers from related fields, etc. With this goal in mind, this introduction describes the basic ideas behind interval computations and behind the applications of interval computations that are surveyed in the book.

1 WHAT ARE INTERVAL COMPUTATIONS?

The Majority Of Real-Life Application Problems Lead To (Constrained Or Unconstrained) Global Optimization

In a typical real-life application problem, we need to find an alternative x that satisfies given constraints and that optimizes the value of a given characteristic $y = f(x)$ (the optimized function $f(x)$ is usually called an *objective function*). For example:

- in *data processing*, we must find a model x that best fits the known data; in this case, $f(x)$ describes how well x fits;

- in *control* , we must find the control strategy x that optimizes the given criterion $f(x)$ (e.g., smoothness of the resulting trajectory);
- in *decision making*, we must find an alternative x from a given set X of possible alternatives that guarantees the best outcome $f(x)$.

From the mathematical viewpoint, all these real-life application problems can be described as (constrained and unconstrained) *global optimization problems.*

In Some Cases, We Must Solve Systems Of Equations And/Or Inequalities

In some real-life problems, there are so many constraints on the problem that even finding a single solution x that satisfies all these constraints is a difficult task. This often happens in design problems; e.g., if we want to design a spacecraft that can fulfill the task of taking photos of a distant planet under restrictions on weight and cost. For these problems, there is no question of a choice between several possible solutions (we are lucky if we have one), and therefore, it makes no practical sense to choose an objective function for this choice. Therefore, such problems are formulated as the problems of satisfying given constraints. Each of these constraints is either an inequality (e.g., "total cost $\leq$ a given amount"), or an equality (e.g., "at a given moment of time, the position of the spacecraft must be exactly a given number of miles above the explored planet"). Therefore, in mathematical terms, these problems are formulated as solution of a system of equations and/or inequalities.

In other cases (e.g., in economic problems), we know the constraints well, but the objective is only informally formulated (e.g., "welfare of the nation"). In these cases, it is difficult to formulate the objective function in precise mathematical terms; therefore, it makes sense to provide the decision-makers with the entire set of the alternatives x that satisfy all given constraints, so that these decision-makers will make the final decision. In such problems, we also need to solve a system of equations and/or inequalities.

Numerical Algorithms Usually Result In Approximate Solutions

There exist many numerical algorithms for solving optimization problems (see, e.g., [17]). Algorithms also exist for solving systems of equations and inequalities.

The majority of these algorithms are iterative, so, when we stop after a certain number of steps, we only get an *approximation* $\tilde{x}$ to the desired solution x.

For Many Practical Cases, We Need Guaranteed Estimates Of The Solution's Accuracy

A natural question is: how good is the approximate solution x? That is, how close is the approximate solution to the desired solution x?

In many practical problems, we need a *guaranteed* solution, i.e., a solution that is guaranteed to be sufficiently close to an optimum (i.e., for which the distance $\rho(\tilde{x}, x)$ between x and $\tilde{x}$ does not exceed a certain given number ε). To get such a guarantee, we definitely need to estimate the closeness $\rho(\tilde{x}, x)$.

Traditional Error Estimating Techniques of Numerical Mathematics Do Not Always Lead to Satisfactory Estimates

In the majority of traditional numerical methods, at each step of the iteration process, we get a number (or, a sequence of numbers) that represents an increasingly better approximation to the solution. After we compute this approximate solution, we use some estimates (e.g., estimates based on Lipschitz constants) to describe how close $\tilde{x}$ is to x. The resulting method does not always lead to good estimates:

- For some numerical algorithms, *no methods are known* yet for estimating the distance $\rho(\tilde{x}, x)$. Designing such an estimation technique for each new algorithm is usually a very complicated problem. Heuristic algorithms are constantly appearing, and there are simply not enough researchers in the numerical analysis community to find estimates for all of them.

- Error estimates of numerical mathematics are usually *overestimating* . One of the reasons for that is that these methods are usually based only on the final result $\tilde{x}$ of the computations. We could get better estimates if we could use more information about the behavior of the function $f(x)$. To get this information, we either need to do more numerical experiments with the function f, or we can use the values that have been computed during the previous computation steps. In both cases, we need *extra computation time*, and this computation time is added to the running time of the numerical algorithm itself, which is often already large.

An Alternative Error Estimation Approach – A Natural Idea

An ideal situation would be if we could estimate the errors of the result not *after* the iteration process (thus, adding time), but *simultaneously* with the iteration process. This is the main idea behind the *interval computations* methodology originally proposed by R. E. Moore [13, 14, 15].

This idea will help us solve the above-mentioned problems:

- First, it will enable us to estimate errors for the approximate solution while computing the solution itself, thus saving *computation time*.
- Second, since the error estimation will take place throughout the entire process of computing $\tilde{x}$, we will be able to use intermediate results in this error estimation and therefore, hopefully, come up with better estimation results (i.e., results that are *less overestimating*).
- Finally, this idea will enable us to easily find estimates for new algorithms: Indeed, it is difficult to compute an error estimate for the result of a lengthy computation. However, each lengthy computation consists of a sequence of elementary steps ($+$, $-$, $\cdot$, $/$, ...). Thus, if we trace this error step-by-step, then this complicated problem is reduced to the problems of estimating the error of a sum (product, etc.) of two previous quantities, when we already know the errors in each quantity.

Why "Interval"?

If we follow the idea described above, then on each computation step, instead of a single (approximate) number $\tilde{r}$, we will compute $\tilde{r}$ *and* an upper bound

Δ for the error of this approximation. In other words, after each computation step, we will get a *pair* $(\tilde{r}, \Delta)$.

If we know $\tilde{x}$ and Δ, this means that the actual value r of the corresponding quantity can take any value from $\tilde{r} - \Delta$ to $\tilde{r} + \Delta$. In other words, this pair means that instead of the (ideal) single value of r, there is a *set* of possible values of r, and this set is an *interval* $\mathbf{r} = [\underline{r}, \overline{r}]$, where $\underline{r} = \tilde{r} - \Delta$ and $\overline{r} = \tilde{r} + \Delta$.

Because of this, the above idea is called *interval computations.*

Basic Formulas Of Interval Arithmetic

For intervals, the above-mentioned step-by-step process works as follows: at each elementary computation step, we perform one of the arithmetic operations $c := a \odot b$, where the two values a and b have already estimated on the previous steps of this algorithm. Since the values a and b have already been estimated by our algorithm, this means that we already have the *intervals* $[\underline{a}, \overline{a}]$ and $[\underline{b}, \overline{b}]$ of possible values of a and b. Since we know that $a \in [\underline{a}, \overline{a}]$ and $b \in [\underline{b}, \overline{b}]$, we can conclude that c belongs to the set of all possible values of $a \odot b$ for such a and b, i.e., to the set

$$\{a \odot b | a \in [\underline{a}, \overline{a}], \text{and} b \in [\underline{b}, \overline{b}]\}.$$

Example. If $\odot = +$, and if we know that $a \in [0.9, 1.1]$ and $b \in [2.9, 3.1]$, then we can conclude that

$$a + b \in \{a + b | a \in [0.9, 1.1] \text{and} b \in [2.9, 3.1]\}.$$

One can easily see that this set is the interval $[3.8, 4.2]$.

In general, the new set (of possible values of c) is called the *result of applying the operation* $\odot$ *to intervals* $[\underline{a}, \overline{a}]$ *and* $[\underline{b}, \overline{b}]$, and it is denoted by

$$[\underline{a}, \overline{a}] \odot [\underline{b}, \overline{b}].$$

For basic arithmetic operations, it is easy to find explicit expressions for the resulting set:

- $[\underline{a}, \overline{a}] + [\underline{b}, \overline{b}] = [\underline{a} + \underline{b}, \overline{a} + \overline{b}]$;
- $[\underline{a}, \overline{a}] - [\underline{b}, \overline{b}] = [\underline{a} - \overline{b}, \overline{a} - \underline{b}]$;
- $[\underline{a}, \overline{a}] \cdot [\underline{b}, \overline{b}] =$
  $[\min(\underline{a}\underline{b}, \underline{a}\overline{b}, \overline{a}\underline{b}, \overline{a}\overline{b}), \max(\underline{a}\underline{b}, \underline{a}\overline{b}, \overline{a}\underline{b}, \overline{a}\overline{b})]$;
- $[\underline{b}, \overline{b}]^{-1} = [1/\overline{b}, 1/\underline{b}]$ if $0 \notin [\underline{b}, \overline{b}]$;
- $[\underline{a}, \overline{a}]/[\underline{b}, \overline{b}] = [\underline{a}, \overline{a}] \cdot [\underline{b}, \overline{b}]^{-1}$.

These operations are called *interval arithmetic*. To these operations, we must add interval analogues of standard functions (exp, sin, max, etc).

For iterative algorithms, we can also take into consideration the fact that, in these algorithms, we repeatedly compute estimates for one and the same quantity. Usually, we use a previous estimate $x^{(k)}$ to compute the next (usually, better) estimate $x^{(k+1)}$. If we use interval operations, then on these two successive steps, we get *intervals* $\mathbf{x}^{(k)}$ and $\mathbf{x}^{(k+1)}$. Since both intervals contain the desired value x, we can conclude that x belongs to the *intersection* of these intervals. Therefore, as an interval that contains x, on the next step of interval computations, we can take not the interval $\mathbf{x}^{(k+1)}$, but the (usually smaller) interval $\mathbf{x}^{(k+1)} \cap \mathbf{x}^{(k+1)}$.

These Formulas Must Be Modified To Take Truncation Errors Into Consideration

The formulas of interval arithmetic are only valid for idealized computers, in which all arithmetic operations are performed exactly. Real-life computers truncate or round the result of multiplication and division, thus adding additional *roundoff* errors . To take these errors into consideration, we must modify the above formulas by using the so-called *directed roundings*, i.e.:

- a rounding $\phi_-(r)$ that always returns a lower bound $\phi_-(r) \le r$, such that $\phi_-(r) \le r$ is a machine number; and
- a rounding $\phi_+(r)$ that always returns an upper bound $\phi_+(r) \ge r$, such that $\phi_+(r) \le r$ is a machine number.

For example, the modified formula for $[\underline{b}, \overline{b}]^{-1}$ is (for $0 \notin [\underline{b}, \overline{b}]$):

$$[\underline{b}, \overline{b}]^{-1} = [\phi_-(1/\overline{b}), \phi_+(1/\underline{b})].$$

This Methodology Can Also Take Into Consideration Uncertainty In The Input Data

In many real-life situations, the input data come from measurements. Measurements are not 100% precise. Therefore, if we have made the measurement with an accuracy Δ, and obtain the measurement result $\tilde{y}$, this means that the actual value y of the measured quantity can take any value from the interval $\mathbf{y} = [\tilde{y} - \Delta, \tilde{y} + \Delta]$. The desired solution x depends on the exact value of y from this interval. Hence, it makes sense to produce the set of all possible solutions x that correspond to all possible values $y \in \mathbf{y}$.

To generate this estimate, we can start with the intervals for input data (instead of the precise values), and then follow interval computations step-by-step.

Brief Summary: The Power Of Interval Computations

The power of interval arithmetic comes from the following facts:

1. The elementary operations and standard functions (such as sin, cos, and exp, found in standard Fortran) can be computed for intervals by using simple formulas and subroutines; and
2. *directed roundings* can be used, so that the images of these operations (e.g. $[3.8, 4.2]$ in the preceding example) *rigorously* contain the *exact* result of the (ideal) computations.

These facts allow

- rigorous enclosure of roundoff error, truncation error, and errors in data;
- computation of rigorous bounds on the ranges of functions.

Naive and Sophisticated Interval Computations

Interval computations were first applied (in the 1960's) *naively*, that is, to existing numerical algorithms. In some cases, reasonable estimates appeared as a result. In other cases, the resulting intervals were too wide to be useful. These overestimates made many researchers and practitioners reluctant to use

interval methods. Many researchers, not familiar with the field, are still under the impression that interval computations are not a useful tool.

However, in the last three decades, many *sophisticated* ideas and algorithm modifications have been proposed that enable replacing the excessively wide intervals of naive interval computations with reasonably narrow ones. In some cases, radically new algorithms emerged that were specifically tailored for interval computations. For a general survey of basic modern interval techniques, see, e.g., [6]; basic interval optimization techniques are described in [7].

As a result of these improvements, modern interval computations have been successfully *applied* to many real-life problems.

2 INTERNATIONAL WORKSHOP ON APPLICATIONS OF INTERVAL COMPUTATIONS: HOW THIS BOOK ORIGINATED

To promote real-life applications of interval computations, an International Workshop on Applications of Interval Computations (APIC'95) was organized in El Paso, Texas, on February 23–25, 1995. For this workshop, 80 researchers from 15 countries (Austria, Brazil, Canada, China, Czech Republic, Denmark, Finland, France, Germany, Mexico, Poland, Russia, Spain, Ukraine, and the USA) submitted papers describing numerous applications of interval computations:

- To *Engineering*:
  - to manufacturing, including:
    - quality control;
    - detection of defects in computer chips;
    - flexible manufacturing;
  - to automatic control, including:
    - control of airplane engines;
    - control of electric power plants;
  - to robotics;

  - to airplane inertial navigation;
  - to civil engineering, including traffic control.
- To *Ergonomics* and *Social Sciences*:
  - to learning curves (that describe how people learn);
  - to project management;
  - to service systems;
  - to sociology.
- To Physics:
  - to laser beams;
  - to particle accelerators;
  - to astrophysics (planet atmospheres);
  - to image processing in radioastronomy.
- To *Geology* and *Geophysics*.
- To *Chemistry*, including:
  - to spectral analysis.
- To *Computer Science and Engineering*:
  - to expert systems;
  - to communication networks, especially computer networks.
- To *Economics*:
  - to planning;
  - to banking.

Extended abstracts of these presentations appeared in [12]. The majority of these papers described specific applications. (Technical details of different specific applications will also appear in the special issue of the international journal *Reliable Computing*.)

In addition to the papers describing technical details of specific applications, we also had several general survey papers, of general interest to the public. It is these surveys that we present in this book.

This book contains surveys of the applications of interval computations, i.e., applications of numerical methods with automatic result verification.

3 GENERAL OPTIMIZATION PROBLEMS

A General Survey

Interval arithmetic is a convenient and effective means of obtaining information used in the place of Lipschitz constants. Such information, combined with bounds on the ranges of functions that interval arithmetic supplies, is is widely recognized as valuable in algorithms for *global optimization*. In this book, a general survey of such algorithms, titled "A Review of Techniques in the Verified Solution of Constrained Global Optimization Problems" [10] is given by Kearfott. In this survey, Kearfott:

- reviews the literature on the subject (including several books);
- outlines the basic techniques (including some of his own experimental ones); and
- gives advice on the use of different techniques.

Optimization Techniques Based On Constraint Propagation

Traditionally , before applying numerical techniques to a problem, we analyze it to formulate a compact and understandable description. Namely:

- First, we describe all the physical properties characterizing the object, and all the equations and inequalities that describe relations between these properties.
- Usually, some of these variables can be expressed in terms of the others. As a result, we can express these variables (called *dependent*) in terms of the remaining few *independent* variables. We can then reformulate all the restrictions in terms of these independent variables.

Thus, we obtain a constrained optimization problem $f(x) \to \max$ with few variables $x = (x_1, ..., x_n)$ and with (often rather complicated) objective function and constraints. (They are complicated because they come from repeated substitution of expressions into other ones, and each such substitution increases the complexity of the resulting expression.)

This reduction is useful, because the running time of existing global optimization techniques grows rapidly with the number of variables, so, the fewer variables we have, the faster we can find the optimum.

The reduction itself, however, is time-consuming. It makes sense if we have many similar problems. In this case, we do this reduction once, for a generic problem, and then use the results of this reduction to save time on all these problems.

There are fields, however, such as robotics and computer-aided geometric design, in which preliminary reduction is not possible. In problems so arising, many variables describe the system: e.g.:

- To describe the state of a robot, we need to know the coordinates and velocities of all its joints, as well as the coordinates of all the points of the geometric environment in which the robot currently operates.
- To characterize a design, we also usually need many design parameters.

Not all of these characteristics are independent: the lengths of the joints restrict their respective positions; design demands impose heavy restrictions on the design parameters so that only few of them remain independent, etc. So, in all these cases, we can apply reduction to get an optimization problem with fewer variables. However, we cannot make these reductions once and forever, because the problems are constantly changing:

- For a manufacturing robot, the problem drastically changes if a new part is added.
- For a design problem, the reduction changes every time we add or delete one of the design requirements. (Such change in design requirements is typical for a design process.)

In these cases, we should solve unreduced optimization problems, i.e., problems in which there are many different variables, but also many constraints that in effect leave only few of the variables independent. Because of the large number of variables, general optimization techniques do not behave well for such problems, so we need new methods.

The corresponding technique is called *constraint propagation*. In this technique, relationships among intermediate quantities in arithmetic expressions in

constraints are used recursively to compute ever-narrower bounds on solution variables.

It turns out that constraint propagation can be *successfully applied* not only to the above-mentioned problems like robotics, in which symbolic reduction is not practical, but also to *general* optimization and equation-solving problems. Often, even if there is a way to reduce the number of constraints, it is actually advantageous to put in more constraints (but make these constraints simpler) and then apply constraint propagation techniques. Thus, constraint propagation is a new powerful general optimization tool.

Several authors have applied the technique in global optimization and nonlinear equation systems codes and software. It turns out that the

In the survey "Solving Optimization Problems with Help of the UniCalc Solver" [21], presented in this book, Semenov describes an integrated interactive environment, available commercially, that uses so-called *sub-definite programming* (a generalization of interval computations to sets more general than intervals) to solve various types of optimization problems.

Two Surveys Of Specific Techniques

In addition to the general surveys of different interval optimization techniques, the book also contains two surveys of interval optimization techniques tailored to problems in two specific fields: manufacturing (Hadjihassan et al [5]) and quantum mechanics (Fefferman and Seco [3]).

Applications To Manufacturing

In "Quality Improvement via The Optimization Of Tolerance Intervals During The Design Stage" [5], Hadjihassan et al use the fundamental numerical analysis considerations described above in manufacturing design. Uncertainties in the input data are encompassed by describing them as intervals. Namely, in [5], Hadjihassan, Walter and Pronzato use a combination of interval and non-interval techniques to solve the following real-life optimization problem: In mass manufacturing, we can usually only reproduce the parameters of the manufactured objects with a certain accuracy δ (typically, 5% or 10%). As a result, when we fix the nominal values x_i of these parameters, we can only guarantee that the actual values of the parameters of the manufactured objects are inside the interval $[x_i - \delta \cdot x_i, x_i + \delta \cdot x_i]$. Hence, even if we choose nominal

design parameters $x = (x_1, ..., x_n)$ that guarantee the desired value y^* of the performance characteristic $y = \eta(x_1, ..., x_n)$, the actual values of y may differ from y^*. The quality $f(x)$ of a design can be characterized, crudely speaking, by the worst case deviation of y from y^*. In Hadjihassan et al [5], interval computations help to solve the problem of finding the optimal (best quality) design, i.e., a design x for which $f(x)$ is minimum.

Applications To Quantum Mechanics

Many optimization problems occur in fundamental physics. Actually, fundamental physical theories are usually formulated not in terms of differential equations (as in Newton's time), but in terms of an appropriate *variational principle* : The characteristics x of the particles and fields must be chosen to optimize the value of an objective function $S(x)$ (called *action*). This use of optimization is not simply a mathematical trick: it has a fundamental justification in quantum mechanics (see, e.g., [4]).

Because of the optimization formulation of fundamental physical theories, numerical optimization techniques, especially interval-type techniques that provide automatic verification, appear to be of great use to physicists.

The majority of unsolved fundamental problems in physics, however, are mainly related to areas such as quantum field theory, (quantum) cosmology, etc, for which the equations are not yet confidently known. Since we are not sure about the equations anyway, there is not much need for a *guaranteed* solution. Therefore, physicists use heuristic techniques, and additional computational efforts needed to estimate the accuracy do not seem to be justified.

There is, however, a class of fundamental problems where the corresponding optimization criterion (and related equations) have been known for over 70 years, and the main problems are computational: these are the problems of non-relativistic quantum mechanics. The fundamental equation was described by Schroedinger as early as 1924; this equation describes all the properties of the simplest atoms (hydrogen and helium) with such a good accuracy that physicists were convinced that what was previously known as chemistry would be replaced by fundamental quantum physics. This did not happen because, to describe an atom with n electrons, we need to find a function of $3n$ variables (by solving an optimization problem or an equivalent differential equation). Even if we take 2 possible values of each of these coordinates, we still need 2^{3n} values only to describe function. For realistic n, this problem is clearly computationally intractable.

To solve this problem, several approximate equations and optimization formulations have been proposed:

- Some of them are computationally easy, but lead to very crude (and thus practically useless) estimates.
- Other approximate equations lead to better estimates, but are still relatively computationally complicated. One such approximation, called *Thomas–Fermi* theory , is considered in the survey "Interval Arithmetic in Quantum Mechanics" [3] by Fefferman and Seco, that is presented in the book.

We are interested in the characteristics of the atoms with large n (because for small n, we can simply solve the exact equations). It is known that for large very n, these characteristics tend to behave in a regular fashion, so, to get a good estimate, we can expand the values of the characteristics into an asymptotic series in $1/n$, and take only a few terms in this expansion. Hence, we need to find *asymptotic formulas* for $n \to \infty$.

Several heuristic optimization methods have been proposed to find the asymptotics of the optimum in this problem. These methods, however, are known to produce good results for some physical problems, but poor estimates for others. Therefore, there is a real need to apply techniques with automatic result verification, and thus get guaranteed estimates on the asymptotics.

Such estimates are obtained in a survey [3] by Fefferman and Seco. Their contribution is thus an example of a computer-assisted proof outside of general global optimization algorithms. Fefferman and Seco explicitly discuss the role of interval arithmetic in establishment of an inequality related to a precise asymptotic formula for the ground state of a non-relativistic atom.

4 GENERAL SYSTEMS OF EQUATIONS AND INEQUALITIES

In this book, a survey of *constraint propagation* techniques for solving such systems is presented. In this technique, sometimes implemented in languages such as Prolog, relationships among intermediate quantities in arithmetic expressions in constraints are used recursively to compute ever-narrower bounds

on solution variables. As we have already mentioned, constraint propagation is increasingly popular in recent years in fields such as robotics and computer-aided geometric design. Systems of equations that it allows us to solve can tell, e.g., whether there is an intersection or non-intersection, e.g.:

- whether after a planned movement the robot will hit the obstacle, or move safely;
- whether a cutting instrument of a manufacturing robot will actually start cutting the desired part;
- whether in a computer design all components of the car's design will actually fit into the car frame (intersection in this case will mean that they won't), etc.

Several authors have applied this technique in general global optimization and nonlinear equation systems codes and software.

This volume contains descriptions of two software systems that allow such computations. We have already mentioned a survey by Semenov [21]. In "Interval Computations on the Spreadsheet" [8], Hyvönen and De Pascale explain an extension of Microsoft Excel that allows interval computations, and constraint propagation in particular.

5 LINEAR INTERVAL PROBLEMS

Why Linear Problems?

In many real-life situations, the general problem (outlined in the beginning of this introduction) can be simplified. A typical situation in which simplification is possible is when we know that the components x_1, .., x_n of the desired solution x are relatively small. Then, we can expand all the terms in all the equations that determine x into power series in x_i, and neglect quadratic and higher order terms.

As a result, we get a problem with:

- a linear objective function $y = f(x) = \sum c_j x_j$, and
- linear constraints: i.e.,
  - linear equalities $\sum a_{ij} x_j = b_j$, and/or
  - linear inequalities $\sum a_{ij} x_j \geq b_i$.

This problem is called a *linearization* of the original non-linear problem.

In other words, in many real-life situations, we must solve linear problems. If all the coefficients a_{ij}, b_i, and c_j of the corresponding linear functions are precisely known, then we have a *linear programming* problem for which many efficient algorithms are known.

If we do not know the objective functions, then, as we have mentioned, we must describe the set X of all the vectors x that satisfy the given linear constraints. This description can also be given by linear programming.

Interval Linear Problems

In many real-life problems, the only source of the values of the coefficients a_{ij} and b_i is measurement. Since measurements are never absolutely precise, as a result of these measurements, we only get *intervals* $\mathbf{a}_{ij}$ and $\mathbf{b}_j$ that contain the (unknown) values of these coefficients. In this case, we are interested in describing the set X of all the vectors x that satisfy the given linear constraints, for *some* $a_{ij} \in \mathbf{a}_{ij}$ and $b_j \in \mathbf{b}_j$.

How To Describe The Solution Of An Interval Linear Problem

For symmetric and non-symmetric a_{ij}, the set X described above is characterized in the paper "The Shape of the Symmetric Solution Set" [1] by Alefeld, Kreinovich, and Mayer, presented in this book. Namely, this set is an an intersection of half-spaces (sets described by linear inequalities) and quadrics (i.e., sets defined by quadratic inequalities).

General Interval Linear Problems Are Hard To Solve

The description of a solution of an interval linear problem provided in Alefeld et al [1] is computationally complicated. A natural question is: is it really a

hard problem and no easy algorithm is possible, or there is a simple algorithm, but we simply have not found it yet?

The answer to this question is given in a paper by Rohn "Linear Interval Equations: Computing Enclosures with Bounded Relative Or Absolute Overestimation is NP-Hard " [19], presented in the book. Namely, Rohn shows that, crudely speaking, there is no feasible (polynomial time) algorithm that computes bounds on the solution sets X to *all* interval linear systems with overestimation less than δ, for any $\delta > 0$.

Existing Algorithms Can Often Solve Interval Linear Problems Efficiently

This general negative result means that, for every algorithm for solving interval linear problems, there are hard cases in which this algorithm either runs too long, or results in a drastic overestimation of X.

In practice, several (heuristic) methods such as interval Gaussian elimination or the interval Gauss–Seidel method work well for many real-life systems. In particular, they work well when the widths of the coefficient intervals are small, and also in many other cases.

Applications To Economics

One case where linear models naturally appear is economics. Namely, the dependency $y = f(x_1, ..., x_n)$ of the amount y of goods produced in an economic sector on the amounts of goods x_i used in production in this production can be described (with a reasonably good accuracy) by a linear function. The corresponding linear model was first proposed by the Nobel laureate economist W. Leontief, and is therefore called a *Leontief model* . A typical problem here is to determine the desired amounts of production for each sector so that all consumer demands are satisfied, and there is no unnecessary production (i.e., all goods are either directly consumed, or used in production of other goods in the chain that leads to consumption). In his pioneering research, Leontief assumed that the coefficients are known precisely. In this case, we have a linear programming problem, that is (in principle) feasible.

In real life, however, the only way to determine the coefficients is from the empirical data. Empirical economic data is not precise. Therefore, instead of

precise values for the coefficients, we have *intervals* of possible values. Thus, in real life, Leontief's problem leads to an interval linear system.

Luckily, in spite of the general computational complexity result, specific interval linear systems stemming from these economic problems are feasible. In this book, a survey of corresponding methods and applications is given by Jerrell in "Applications of Interval Computations to Regional Economic Input-Output Models" [9]. As a case study, Jerrell obtains meaningful bounds for the impact of Northern Arizona University on the economy of Coconino County, Arizona, and on the state of Arizona in general.

6 INTERVAL COMPUTATIONS CAN ALSO HANDLE POSSIBLE ADDITIONAL INFORMATION ABOUT THE INPUT DATA

We have already mentioned that interval computations can naturally handle situations in which, instead of the exact values of the input parameters p, intervals $\mathbf{p}$ of possible values of these parameters are known.

In many real-life cases, in addition to these intervals, we have additional information about p:

- In some cases, we know not only the *interval* of possible values of p, but we also have some information about the *probabilities* of different values. If we know all the probabilities for all possible values of all parameters, then we can apply traditional statistical techniques. However, if we only know some probabilities for some of the parameters, and only interval information for the others, we need a special technique for combining these two types of information. In this book, a survey of such techniques and their applications is given by Berleant "Automatically Verified Arithmetic on Probability Distributions and Intervals" [2].
- In some cases, in addition to the interval in which the value of p is guaranteed to lie, we have experts who produce narrower intervals to which, they claim, the actual values "most certainly" belong. In this case, in addition to the interval of "definitely possible" values of x (produced by interval computation techniques), it is desirable to supply the user with

narrower intervals that result from the experts' narrower input intervals. (These new intervals will be correct enclosures if the experts did not err in their estimates.) In this book, this problem is described and analyzed in a survey "Nested Intervals and Sets: Concepts, Relations to Fuzzy Sets, and Applications" [16] by Nguyen and Kreinovich. It turns out that the resulting formalism is very close to the formulas proposed in so-called *fuzzy logic* (often without rigorous justification). Namely, to get these formulas, as a "fuzzy truth value" $\mu(A)$ of a statement A, we must take, crudely speaking, the ratio $\mu(A) = N(A)/N$ of those experts ($N(A)$) who believe A to be true to the total number N of the experts asked. Thus, this version of interval computation leads to a new justification of the formulas of fuzzy logic.

Two other surveys from this book describe further modifications of this idea:

- The first modification is described by Kohout and Bandler in "Fuzzy Interval Inference Utilizing the Checklist Paradigm and BK-Relational Products" [11]. This modification is based on the fact that we can only ask so many questions to the experts. In particular, if we have, say, n statements $A_1, ..., A_n$ of interest, then we can ask an expert's opinion about these n statements, maybe about some of their logical combinations (of the type $A_1 \& A_2$), but *not* about all $> 2^n$ possible logical combinations of these statements. As a result, there may be some statements A_i and A_j for which we only know the truth values $\mu(A_i)$ and $\mu(A_j)$, but not $\mu(A_i \& A_j)$. Suppose that we are interested in knowing the degree of belief in $A_i \& A_j$. If, e.g., $\mu(A_i) = \mu(A_j) = 0.5$. Then there are two possibilities:
  - It could be that the same experts believe in A_i as in A_j. In this case, $\mu(A_i \& A_j) = \mu(A_i) = 0.5$.
  - It can also be that the experts who believe in A_i are exactly those who *do not* believe in A_j. In this case, $\mu(A_i \& A_j) = 0$.

  In general, for a given $\mu(A_i)$ and $\mu(A_j)$, we have an *interval* of possible values of $\mu(A_i \& A_j)$. If we already have intervals of possible values for $\mu(A)$ and $\mu(B)$, then we need an interval operation to compute the interval of possible values of $\mu(A \& B)$. Here, interval computations come in handy. As a case study, Kohout and Bandler describes an implementation of this idea in an expert medical system *Clinaid*.
- Another modification is described by Rocha in "Computing Uncertainty in Interval Based Sets" [18]. The underlying idea is that experts are often inconsistent. Therefore, instead of a single small interval of possible values, we get several (often non-intersecting) intervals that

supposedly describe the same quantity. Some intervals are supported by more experts, some by less. To describe the degree of support for each interval $\mathbf{x}$, we can use a ratio $N(\mathbf{x})/\mathbf{N}$ similar to that described above. Since each expert selects some interval, these ratios add up to 1 and can thus be mathematically interpreted as (subjective) probabilities. Therefore, to describe the expert's opinions, we must have a *probability distribution on the set of all intervals.* An interval is a particular case of this distribution, when only one probability is different from 0. In Rocha [18], standard operations of interval arithmetic are extended to these new objects . Applications to knowledge representation in expert systems are described.

7 SOFTWARE AND HARDWARE SUPPORT FOR INTERVAL COMPUTATIONS

If interval methods are so good, why aren't they widely applied? One of the main reasons is lack of easily available software. Now, people rarely program "from scratch": they try to use existing software as much as possible. Therefore, to make interval computations usable, the computations must be supported by *software* packages.

This brings us to the next issue: a software package will not be used if it is not sufficiently fast. Everyone is interested in buying a faster computer, and "a faster computer" means that computer on which typical "benchmarking" programs run faster. The bulk of existing software does not use intervals at all; therefore, when designing new computers, computer engineers try to speed up the operations with numbers that are most frequently used in the usual programs. As a result, operations with numbers are fast, while elementary operations with intervals (that form the basis for interval computations) are much slower. Because of that, experts in interval computations have urged the development of faster, *hardware*-oriented systems for interval computations.

In their survey "Software and Hardware Techniques for Accurate, Self-Validating Arithmetic" [20], Schulte and Swartzlander review some of the available programming language extensions , packages and problem solving environments for interval computations. They then discuss hardware design

alternatives, and propose a specific hardware design and software interface for variable precision interval arithmetic.

Finally, in "Stimulating Hardware and Software Support for Interval Arithmetic" [22], Walster presents his view of why many hardware and software vendors do not presently provide interval computations as integral parts of their products. He then discusses steps the community of interval computations experts should take to make such capabilities more widely available.

Acknowledgements

This work was supported in part by the *National Science Foundation (NSF)*, through grants CCR-9203730, CDA-9015006, and EEC-9322370, and by the *National Aeronautics and Space Administration (NASA)*, through grant No. NAG 9-757.

REFERENCES

[1] G. Alefeld, V. Kreinovich, and G. Mayer, "The Shape of the Symmetric Solution Set," *This Volume.*

[2] D. Berleant, "Automatically Verified Arithmetic on Probability Distributions and Intervals", *This Volume.*

[3] C. L. Fefferman and L. A. Seco, "Interval Arithmetic in Quantum Mechanics", *This Volume.*

[4] R. P. Feynman, R. B. Leighton, and M. L. Sands, *The Feynman Lectures On Physics*, Addison-Wesley, Redwood City, CA, 1989.

[5] S. Hadjihassan, E. Walter, and L. Pronzato, "Quality Improvement via The Optimization Of Tolerance Intervals During The Design Stage," *This Volume.*

[6] R. Hammer, M. Hocks, U. Kulisch, D. Ratz, *Numerical toolbox for verified computing. I. Basic numerical problems*, Springer Verlag, Heidelberg, N.Y., 1993.

[7] E. R. Hansen, *Global optimization using interval analysis*, Marcel Dekker, N.Y., 1992.

[8] E. Hyvönen and S. De Pascale, "Interval Computations on the Spreadsheet", *This Volume.*

[9] M. E. Jerrell, "Applications of Interval Computations to Regional Economic Input-Output Models," *This Volume.*

[10] R. B. Kearfott, "A Review of Techniques in the Verified Solution of Constrained Global Optimization Problems," *This Volume.*

[11] L. J. Kohout and W. Bandler, "Fuzzy Interval Inference Utilizing the Checklist Paradigm and BK-Relational Products", *This Volume.*

[12] V. Kreinovich (ed.), *Reliable Computing*, 1995, Supplement (Extended Abstracts of APIC'95: International Workshop on Applications of Interval Computations, El Paso, TX, Febr. 23–25, 1995).

[13] R. E. Moore, *Automatic error analysis in digital computation*, Lockheed Missiles and Space Co. Technical Report LMSD-48421, Palo Alto, CA, 1959.

[14] R. E. Moore and C. T. Yang, *Interval analysis*, Lockheed Missiles and Space Co. Technical Report LMSD-285875, Palo Alto, CA, 1959.

[15] R. E. Moore, *Interval analysis*, Prentice Hall, Englewood Cliffs, NJ, 1966.

[16] H. T. Nguyen and V. Kreinovich, "Nested Intervals and Sets: Concepts, Relations to Fuzzy Sets, and Applications", *This Volume.*

[17] P. M. Pardalos and J. B. Rosen, *Constrained Global Optimization: Algorithms and Applications*, Springer-Verlag, New York, 1987.

[18] L. M. Rocha, "Computing Uncertainty in Interval Based Sets", *This Volume.*

[19] J. Rohn, "Linear Interval Equations: Computing Enclosures with Bounded Relative Or Absolute Overestimation is NP-Hard," *This Volume.*

[20] M. J. Schulte and E. E. Swartzlander, Jr., "Software and Hardware Techniques for Accurate, Self-Validating Arithmetic," *This Volume.*

[21] A. L. Semenov, "Solving Optimization Problems with Help of the UniCalc Solver," *This Volume.*

[22] G. W. Walster, "Stimulating Hardware and Software Support for Interval Arithmetic," *This Volume.*