7-1998

Beyond Interval Systems: What is Feasible and What is Algorithmically Solvable?

Vladik Kreinovich
The University of Texas at El Paso, vladik@utep.edu

Comments:
Technical Report: UTEP-CS-98-18
Published in: Panos M. Pardalos (ed.), *Approximation and Complexity in Numerical Optimization: Continuous and Discrete Problems*, Kluwer, Dordrecht, 1999, pp. 364-379.

Beyond Interval Systems: What Is Feasible and What Is Algorithmically Solvable?

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

Abstract

In many real-life applications of interval computations, the desired quantities appear (in a good approximation to reality) as a solution to a system of interval linear equations. It is known that such systems are difficult to solve (NP-hard) but still algorithmically solvable. If instead of the (approximate) interval linear systems, we consider more realistic (and more general) formulations, will the corresponding problems still be algorithmically solvable? We consider three natural generalizations of interval linear systems: to conditions which are more general than linear systems, to multi-intervals instead of intervals, and to dynamics (differential and difference equations) instead of statics (linear and algebraic equations). We show that the problem is still algorithmically solvable for non-linear systems and even for more general conditions, and it is still solvable if we consider linear or non-linear systems with multi-intervals instead of intervals. However, generalized conditions with multi-interval uncertainty are already algorithmically unsolvable. For dynamics: difference equations are still algorithmically solvable, differential equations are, in general, unsolvable.

1 Informal Introduction: Why Interval Systems, and Why Go Beyond Interval Systems

Why systems. In many real-life situations, we are interested in the values of physical quantities $x_1, \ldots, x_n$ which are difficult (or even impossible) to measure directly: e.g., we cannot directly measure the distance to a star, or the amount of oil in a well. Since me cannot measure x_j *directly*, we have to measure them *indirectly*, by first measuring some auxiliary quantities $y_1, \ldots, y_m$ which are, in a known way, related to the desired quantities x_j, and then using the known relations to reconstruct x_j from the measured values $\widetilde{y}_k$. For example, to measure the distance to a star of a known spectral type, we measure its visible brightness and thus, reconstruct the distance; to measure the amount of oil, we, e.g., place ultrasonic sources in the well and measure the responses, etc.

In many practical cases, the relations between x_j and y_k take the form of explicit equations $f_i(x_1, \ldots, x_m, y_1, \ldots, y_m) = 0$ (usually, with polynomial f_i); so, to reconstruct x_j, we must solve the corresponding system of equations with n unknowns $x_1, \ldots, x_n$.

Why linear systems. If we know the exact values of y_i, we get a system of equations $F_i(x_1, \ldots, x_n) = 0$ with unknowns $x_1, \ldots, x_n$. Often, we know *approximate* values $x_j^{(0)}$ of the desired quantities; in this case, all we need to know is the *difference* $\Delta x_j = x_j - x_j^{(0)}$ between the actual value x_i and the known approximation $x_j^{(0)}$. Substituting $x_j = x_j^{(0)} + \Delta x_j$ into the equations $F_i = 0$, we get the new equations $G_i(\Delta x_1, \ldots, \Delta x_n) = 0$ with new unknowns $\Delta x_1, \ldots, \Delta x_n$. If the known approximate values are accurate enough, then the differences Δx_j are small; hence, we can expand the functions G_i into Taylor series and ignore terms which are quadratic in Δx_j (or of higher order). Thus, we end up with a system of *linear* equations $\sum a_{ij} \cdot \Delta x_j = g_i$.

There exist feasible algorithms for solving linear systems. Since the goal of this paper is to analyze computational complexity and algorithmic solvability of different problems, we must mention that linear systems are easy to solve: known algorithms for solving these systems are feasible (polynomial-time), i.e., their running time is bounded by a polynomial of the length (= bit size) of the input (see, e.g., [1]).

Why interval systems. As we have mentioned before, if we know the *exact* values of y_i, then after substituting these values into the original polynomial equations $f_i(x_1, \ldots, x_n, y_1, \ldots, y_m) = 0$, we get a

system of polynomial equations $F_i(x_1, \ldots, x_n) = 0$, in which the coefficients of each polynomial F_i are functions of y_j.

In real life, measurements are not 100% accurate, so after measuring each quantity y_i, we do not get the *exact* value of y_i; we often get only the *interval* $\mathbf{y}$ of possible values of y_i. As a result, instead of a system of polynomial equations with *exactly* known coefficients, we have a system of polynomial equations in which we only know the *intervals* of possible values of these coefficients. In particular, for a linear system, we get an interval linear system $\sum \mathbf{a}_{ij} \cdot \Delta x_j = \mathbf{g}_i$, with intervals $\mathbf{a}_{ij}$ and $\mathbf{g}_i$, and we are interested in describing the set of all possible values Δx_j, i.e., the set of all possible solutions of linear systems with $a_{ij} \in \mathbf{a}_{ij}$ and $g_i \in \mathbf{g}_i$. In particular, for each of the variables Δx_j, we are interested in the smallest and largest possible values of Δx_j.

Interval linear systems are algorithmically solvable, but hard to solve. It is know that interval linear systems are algorithmically solvable: there exists an algorithm for finding the desired smallest and largest values of x_j (see, e.g., [7]); such an algorithm can be, e.g., obtained as a particular case of Tarski-Seidenberg algorithm (see e.g., [7]; later in this paper, we will describe this algorithm in some more detail).

The above algorithm takes a very long time; namely, its worst-case running time increases exponentially (or even faster) with the length (= bit size) of the input. This complexity is caused not only by the inefficiency of this algorithm, but by the complexity of the problem itself: this problem is known to be NP-hard (see, e.g., [6, 8, 9, 7]). Crudely speaking, NP-hard means that under a natural hypothesis (P≠NP) in which computer scientists believe, every algorithm for solving this problem requires, in the worst case, at least exponential time (for precise definitions, see, e.g., [3, 7]). Since, e.g., 2^{300} is larger than the lifetime of the Universe divided by the smallest known physically possible time quantum, this worst-case exponential time means that for every algorithm, there exist reasonable-size problems (e.g., of size $n = 300$), for which this algorithm will, for all practical purposes, never find the solution.

Beyond linear interval systems: what is feasible and what is algorithmically solvable? Linear interval systems are already hard to solve, but, as we have indicated before, they are often simply *approximations* to the actual systems. It is therefore desirable to find out how hard are these "actual" systems. We cannot expect these more general systems to be easier to solve than interval linear ones, but at least we would like to know whether they are still algorithmically solvable, i.e., whether there still exist algorithms for solving such systems (maybe algorithms with fast-growing worst-case running times). This is what we plan to analyze in this paper.

How can *actual* conditions on the unknown values x_j be different from a system of interval linear equations?

- First, *equations* may be more complicated than linear ones, i.e., we can have *non-linear* interval systems, i.e., polynomial systems of the type $F_i(x_1, \ldots, x_n) = 0$. In [7], we have shown even for *exact* coefficients, such systems are NP-hard (but algorithmically solvable), and that they remain algorithmically solvable for polynomial systems with interval coefficients.

- Second, we may have more general conditions than simply a system of equations. For example, when we formulate a system of equations, we assume that all measurements are reliable. It can happen that some measurement results can be way off. In such cases, the equations which describe the relation between these measurement results and the desired values x_j, may be false. If we know that a certain portion (e.g., 1%) of equations may be wrong, we may want to look for the set of all possible solutions of all systems obtained from the original one by deleting this portion (1%) of equations. How difficult is this problem? Is it still solvable? What if we have even more complicated conditions, e.g., if we know that the desired values x_j must satisfy some additional property?

- Third, the uncertainty in y_i (and thus, the uncertainty in the coefficients) can be of a more general type than simply an interval of possible values:

  - we may have an *infinite intervals*: e.g., if we know the the resistance R is between 0 and 0.01 Ohm, we can thus conclude that the possible values of the conductivity $c = 1/R$ form an infinite interval $[100, \infty)$;
  - we may have a *multi-interval*, i.e., a finite union of intervals; for example, if we know that v^2 belongs to the interval $[1, 4]$, then possible values of v form a multi-interval $[-2, -1] \cup [1, 2]$.

- Finally, we may want to take into consideration that the above description of the relations between x_j and y_k is *static* (not depending on time), while in reality, this dependence (and the corresponding

conditions on x_j) may be time-dependent; in precise terms, instead of a system of *algebraic* equations, we may have a system of *differential* or *difference* equations.

Of these four directions, the first (as we have already mentioned) is already analyzed in [7]; in this paper, we will complete this analysis by analyzing the questions of feasibility and algorithmic solvability in the remaining three directions.

2 Important Particular Case of Generalized Conditions: Systems of Equations, Some of Which May Be Wrong

Definition 1. *Let $\varepsilon > 0$ be a real number, and let a system of N equations, with unknowns $x_1, \ldots, x_n$ be given.*

- *We say that a vector $(x_1, \ldots, x_n)$ is an $(1-\varepsilon)$-solution to the given system if it satisfies at least $(1-\varepsilon)\cdot N$ of these equations.*
- *We say that the system is $(1-\varepsilon)$-consistent if it has a $(1-\varepsilon)$-solution.*

Comment. When a system is given, a natural question is: is this system $(1-\varepsilon)$-consistent? If it is, then it is natural, for each of the unknowns x_j, to find the interval of its possible values, i.e., the interval $[\underline{x}_j, \overline{x}_j]$ formed by the $\underline{x}_j = \inf x_j$ and $\overline{x}_j = \sup x_j$, where inf and sup are taken over all possible $(1-\varepsilon)$-solutions. It turns out that both problems are NP-hard:

Proposition 1. *Let $\varepsilon = p/q$ be an arbitrary rational number from the interval $(0, 0.5)$. Then, the following two problems are NP-hard:*

- *Given a system of N linear equations (with rational coefficients), check whether this system is $(1-\varepsilon)$-consistent.*
- *Given an $(1-\varepsilon)$-consistent system of N linear equations, and an integer j from 1 to n, find the smallest and the largest possible values of x_j for all possible $(1-\varepsilon)$-solutions.*

Comment. For readers' convenience, all the proofs are placed in the special proofs section.

3 General Case of Interval Conditions

Motivations. How can we describe general conditions? If we know the *exact* equations $F_i(x_1, \ldots, x_n) = 0$, then the desired solution as a vector $(x_1, \ldots, x_n)$ which satisfies all these equations, i.e., for which the following formula is true:

$$F_1(x_1, \ldots, x_n) = 0 \,\&\, F_2(x_1, \ldots, x_n) = 0 \,\&\, \ldots \,\&\, F_N(x_1, \ldots, x_n) = 0.$$

If we only know the *intervals* $\mathbf{y}_i = [\underline{y}_i, \overline{y}_i]$ of possible values of the quantities y_i, then we are interested in finding all possible solutions which correspond to different $y_i \in \mathbf{y}_i$, i.e., we are interested in the set of all vectors $x_1, \ldots, x_n$ which satisfy the following property:

$$\exists y_1 \ldots \exists y_m \,(y_1 \in \mathbf{y}_1 \,\&\, \ldots \,\&\, y_m \in \mathbf{y}_m \,\&$$

$$f_1(x_1, \ldots, x_n, y_1, \ldots, y_m) = 0 \,\&\, \ldots \,\&\, f_N(x_1, \ldots, x_n, y_1, \ldots, y_m) = 0).$$

In different practical problems, we can have more complicated conditions: e.g., in control, we may be interested in finding the control values which stabilize the given system for *all* possible values of the parameters within the given interval; in optimal control, we may look for the control which is not only guaranteed to stabilize, but which is also the best (in some precise sense) among all the stabilizing controls, etc. Different formulations of this type were analyzed by Shary (see, e.g., [13]) who showed that the conditions corresponding to these problems can be described by adding quantifiers "for all" and "exists" to elementary formulas. Crudely speaking, if we are interested in the set of *possible* values of y, we are interested in values y for which $\exists x_j$ such that the given equations are true; if we want a control y that leads to stability for *all* possible values x_j, we use a universal quantifier $\forall x_j$.

Let us formalize this idea:

Definition 2. *By a generalized interval condition, we mean a formula in the following language* $L_{\rm int}$*:*

- *We start with variables* x*,* y*,* z*, ..., that run over real numbers, variables* $\mathbf{x}, \mathbf{y}, \ldots$*, that run over intervals, and with all rational numbers* p/q *as constants.*
- *From variables for real numbers and constants, we can form expressions by applying addition and multiplication. For example,* $x \cdot x + y \cdot y$*, or any polynomial expression* $P(x_1, \ldots, x_n)$*, is an expression in this sense.*
- *From expressions* t*,* t'*, ..., we can form elementary formulas of the type* $t = t'$*,* $t \neq t'$*,* $t > t'$*,* $t < t'$*,* $t \leq t'$*, and* $t \geq t'$*. For example,* $x \cdot x + y \cdot y = z \cdot z$ *or* $F_i(x_1, \ldots, x_n) = 0$ *are elementary formulas in our language.*
- *We also consider elementary formulas of the type* $t \in \mathbf{x}$*, where* t *is an expression, and* $\mathbf{x}$ *is an interval variable.*
- *From elementary formulas, we can form formulas by applying logical connectives* $\&$ *("and"),* $\vee$ *("or"),* $\rightarrow$ *("implies"),* $\leftrightarrow$ *("equivalent"),* $\neg$ *("not"), and quantifiers* $\forall x$*,* $\exists x$*,* $\forall \mathbf{x}$*, and* $\exists \mathbf{x}$*.*
- *A formula is called a generalized interval condition if its only free variables are variables* $x_1, \ldots, x_n$ *for real numbers (i.e., if all interval variables are bound by quantifiers).*

Comment. In this definition, the only elementary formulas involving intervals were formulas $t \in \mathbf{x}$. In principle, we can consider other interval-related elementary formulas like $\mathbf{x} \subseteq \mathbf{y}$, or $\mathbf{x} + \mathbf{y} = \mathbf{z}$; however, one can easily see that allowing these new elementary formulas will not change the definition of a generalized interval conditions, because these formulas can be reformulated in terms of already existing ones:

- by definition of a subset, $\mathbf{x} \subseteq \mathbf{y}$ is equivalent to $\forall z\, (z \in \mathbf{x} \rightarrow z \in \mathbf{y})$;
- by definition of the interval sum, $\mathbf{x} + \mathbf{y} = \mathbf{z}$ is equivalent to

$$\forall x \forall y \forall z\, ((z \in \mathbf{z}) \leftrightarrow \exists x \exists y\, (x \in \mathbf{x} \,\&\, y \in \mathbf{y} \,\&\, z = x + y)).$$

Definition 3.

- *We say that real numbers* $x_1, \ldots, x_n$ *form a solution of a generalized interval condition* $F(x_1, \ldots, x_n)$ *if, after substituting these numbers into a formula, we get a true statement.*
- *We say that a generalized interval condition is consistent if it has a solution.*

Comment. Since interval linear equations are a particular case of generalized interval conditions, and solving interval linear equations is NP-hard, solving generalized interval conditions is also an NP-hard problem. The question is: is it algorithmically solvable? Our answer is: Yes. This result is not completely trivial, because, as we will see in the following sections, for *multi-intervals*, a similar problem becomes algorithmically unsolvable.

Proposition 2. *There exists an algorithm which, given a generalized interval condition with* n *real variables:*

- *checks whether this condition is consistent, and*
- *if the condition is consistent, returns, for every* j *from 1 to* n*, the smallest* $\underline{x}_j$ *and the largest* $\overline{x}_j$ *of values of* x_j *for all possible solutions of this condition.*

4 Systems of Equations (and Generalized Conditions) under Multi-Interval Uncertainty

Definition 4.

- *By a generalized interval, we mean an open, closed, semi-open, or infinite interval, i.e., one of the following sets:* $[a,b]$, (a,b), $(a,b]$, $[a,b)$, $[a,\infty)$, (a,∞), $(-\infty,a]$, $(-\infty,a]$, *and* $(-\infty,\infty)$.
- *By a multi-interval, we mean a finite union of generalized intervals.*
- *By a multi-interval algebraic system, we mean a system of* N *equations* $f_i(x_1,\ldots,x_n,y_1,\ldots,y_m)=0$, $1\le i\le N$, *where* f_i *are polynomials with rational coefficients, together with multi-intervals* $\mathbf{y}_k$, $1\le k\le m$. *The variables* $x_1,\ldots,x_n$ *are called unknowns.*
- *We say that a vector* $(x_1,\ldots,x_n)$ *is a solution to a multi-interval algebraic system if there exist* $y_k\in\mathbf{y}_k$ *for which all* N *equations* $f_i=0$ *are true.*
- *We say that a multi-interval linear system is consistent if it has a solution.*

Proposition 3.

- *There exists an algorithm which, given a multi-interval algebraic system, checks whether this system is consistent.*
- *For every consistent multi-interval algebraic system with unknowns* $x_1,\ldots,x_n$, *and for every* j *from 1 to* n, *the set of values* x_j, *corresponding to different solutions* $(x_1,\ldots,x_n)$ *to this system, is a multi-interval.*
- *There exists an algorithm which, given a multi-interval algebraic system with unknowns* $x_1,\ldots,x_n$, *and an integer* j *from 1 to* n, *returns the multi-interval of values* x_j *corresponding to different solutions to this system.*

Comment. In other words, for each j, we can compute the set $\mathbf{x}_j=$

$$\{x_j\,|\,\exists x_1\ldots\exists x_{j-1}\,\exists x_{j+1}\ldots\exists x_n\,((x_1,\ldots,x_n)\text{ form a solution to the system})\}.$$

For more general conditions, computing solutions is algorithmically undecidable. Namely, we can describe *generalized multi-interval conditions* as formulas from the language $L_{\rm mult}$ which is defined as in Definition 2, with the only exception that instead of variables for intervals, we now have variables for *multi-intervals.*

Proposition 4.

- *No algorithm is possible for checking whether a generalized multi-interval condition is consistent or not.*
- *No algorithm is possible which would return, for each consistent generalized multi-interval condition, a vector* $(x_1,\ldots,x_n)$ *which satisfies this condition.*

The results from Propositions 2–4 (as well as the previously known results, cited in [7]) can be represented by the following table:

	(Exact) Real numbers	Intervals	Multi-intervals
Linear systems	Feasible	Algorithmically solvable but NP-hard	Algorithmically solvable but NP-hard
Polynomial systems	Algorithmically solvable but NP-hard	Algorithmically solvable but NP-hard	Algorithmically solvable but NP-hard
Generalized conditions	Algorithmically solvable but NP-hard	Algorithmically solvable but NP-hard	Not algorithmically solvable

5 Dynamic Systems: Differential and Difference Equations

In interval linear systems, we do not take dynamics into consideration. What if we do? Then, instead of a system, we get a system of difference or differential equations. If we know the exact initial conditions, then the problem of solving a differential equation is, given the initial state $s(T_0)$, to compute the state $s(T)$ at some future moment of time T. With interval uncertainty, we only have an interval information about the initial state, and we may be able to predict only the interval information about the resulting state. Hence, we get the following problem:

Definition 5. *Let n be a positive integer.*

- *By a state, we mean a tuple $s = (x_1, \ldots, x_n)$ of n real numbers.*
- *By a interval state, we mean a tuple $\mathbf{s} = (\mathbf{x}_1, \ldots, \mathbf{x}_n)$ of n generalized intervals.*
- *We say that a state $s = (x_1, \ldots, x_n)$ is consistent with the interval state $\mathbf{s} = (\mathbf{x_1}, \ldots, \mathbf{x_n})$ if $x_i \in \mathbf{x}_i$ for all i from 1 to n.*
- *By a polynomial dynamical system, we mean a tuple $(n, P_1, \ldots, P_n)$ of n polynomials of n variables $x_1, \ldots, x_n$.*
- *Let $T_0 < T$ be two integers. We say that a function $s(t) = (x_1(t), \ldots, x_n(t)$ which maps integers t from T_0 to T to states is a discrete-time solution of the dynamical system if for every t from T_0 to $T - 1$, and for every i, $x_i(t + 1) = P_i(x_1(t), \ldots, x_n(t))$.*
- *Let $T_0 < T$ be two rational numbers. We say that a function $s(t) = (x_1(t), \ldots, x_n(t)$ from real numbers $t \in [T_0, T]$ to states is a continuous-time solution of the dynamical system if for every t and i, $\dot{x}_i(t) = P_i(x_1(t), \ldots, x_n(t))$ (where $\dot{x}_i(t)$ denotes time derivative).*
- *By the problem of solving a system of difference equations under interval uncertainty, we mean the following problem: given a polynomial dynamical system, two integers $T_0 < T$, and two interval states $\mathbf{s}^{(0)}$ and $\mathbf{s}$, check whether the given dynamical system has a discrete-time solution for which $s(T_0)$ is consistent with $\mathbf{s}^{(0)}$, and $s(T)$ is consistent with $\mathbf{s}$.*
- *By the problem of solving a system of differential equations under interval uncertainty, we mean the following problem: given a polynomial dynamical system, two rational numbers $T_0 < T$, and two interval states $\mathbf{s}^{(0)}$ and $\mathbf{s}$, check whether the given dynamical system has a continuous-time solution for which $s(T_0)$ is consistent with $\mathbf{s}^{(0)}$, and $s(T)$ is consistent with $\mathbf{s}$.*

Proposition 5.

- *The problem of solving a system of difference equations under interval uncertainty is algorithmically solvable.*
- *The problem of solving a system of differential equations under interval uncertainty is not algorithmically solvable.*

Comment. Whether the problem is algorithmically solvable or not depends on whether we consider discrete or continuous *time*. In [7], it is shown, in essence, that if we consider discrete *space* (i.e., each of the variables x_i which only take integer values), then the problem also becomes algorithmically unsolvable. Thus, the case of continuous space and discrete time is the only algorithmically solvable case. We can express both results in a following table:

	Discrete time	Continuous time
Discrete space	Not algorithmically solvable	Not algorithmically solvable
Continuous space	Algorithmically solvable	Not algorithmically solvable

6 Proofs

Proof of Proposition 1. Let us first prove NP-hardness of checking $(1-\varepsilon)$-consistency. To prove NP-hardness of this problem, we will reduce one of the problems which are already known to be NP-hard to our problem; namely, we will reduce the following *PARTITION* problem: Given n integers $s_1, \ldots, s_n$, check whether there exist values $x_1, \ldots, x_n \in \{-1, 1\}$ for which $s_1 \cdot x_1 + \ldots + s_n \cdot x_n = 0$ (see, e.g., [3, 7]). Let us show that if we can solve our original problem, then we can solve every instance of *PARTITION* problem as well. Indeed, let an instance of the *PARTITION* problem be given. Then, we can take $N = q \cdot n$, and the following system of equations:

- an equation $s_1 \cdot x_1 + \ldots + s_n \cdot x_n = 0$ repeated $(1-2\varepsilon) \cdot N = (q-2p) \cdot n$ times;
- for each j from 1 to n, the equations $x_j = 1$ and $x_j = -1$, each repeated p times.

For each j, we have $2p$ equations; therefore, the total number of equations corresponding to all j is equal to $n \cdot (2p) = 2\varepsilon \cdot N$. So, totally, we indeed have $(1-2\varepsilon) \cdot N + 2\varepsilon \cdot N = N$ equations.

For each j, at most one of the equations $x_j = 1$ and $x_j = -1$ can be true; this means that in all possible cases, for each j, at least q equations corresponding to this j are false. Thus, for all j from 1 to n, at least $n \cdot 2q = N \cdot \varepsilon$ equations are false. Hence, for every vector x_j, at most $(1-\varepsilon) \cdot N$ equations are true. Thus, the only way to make *at least* $(1-\varepsilon) \cdot N = N - \varepsilon \cdot N$ equations true is to make sure that exactly $\varepsilon \cdot N$ equations are true. This means that for each j, one of the equations $x_j = 1$ or $x_j = -1$ is true (i.e., that $x_j \in \{-1, 1\}$), and that the equation $s_1 \cdot x_1 + \ldots + s_n \cdot x_n = 0$ is true. In other words, this means that $x_1, \ldots, x_n$ form a solution to the *PARTITION* problem.

Vice versa, any solution to the *PARTITION* problem satisfies $(1-\varepsilon) \cdot N$ equation and is, thus, a $(1-\varepsilon)$-solution to our system of equations. Thus, our system is $(1-\varepsilon)$-consistent if and only if the original instance of the *PARTITION* problem has a solution. The reduction is completed. Thus, the problem of checking $(1-\varepsilon)$-consistency is indeed NP-hard.

Let us now prove that the problem of computing $\underline{x}_j$ and $\overline{x}_j$ is also NP-hard. We will reduce the same *PARTITION* problem to our new problem. For any instance of the *PARTITION* problem, we will design the following system of $N = q \cdot (n+1)$ equations with $n+1$ unknowns $v_1, \ldots, v_{n+1}$:

- an equation $s_1 \cdot v_1 + \ldots + s_n \cdot v_n + s_{n+1} \cdot v_{n+1} = s_{n+1}$ repeated $(1-2\varepsilon) \cdot N = (q-2p) \cdot (n+1)$ times, where we denoted $s_{n+1} = -0.5 \cdot (s_1 + \ldots + s_n)$; and
- for each j from 1 to $n+1$, the equations $v_j = 1$ and $v_j = -1$, each repeated p times.

For each j, we have $2p$ equations; therefore, the total number of equations corresponding to all j is equal to $(n+1) \cdot (2p) = 2\varepsilon \cdot N$. So, totally, we indeed have $(1-2\varepsilon) \cdot N + 2\varepsilon \cdot N = N$ equations.

This system is $(1-\varepsilon)$-consistent because it has a $(1-\varepsilon)$-solution $v_1 = \ldots = v_n = 1$, $v_{n+1} = -1$. Let us show that for this system, $\overline{v}_{n+1} = 1$ or $\overline{v}_{n+1} = -1$, and $\overline{v}_{n+1} = 1$ if and only if the original instance of the *PARTITION* problem has a solution. Indeed, as in the first part, for any $(1-\varepsilon)$-solution, for each j, we must have $v_j = 1$ or $v_j = -1$, and we must also have $s_1 \cdot v_1 + \ldots + s_n \cdot v_n + s_{n+1} \cdot v_{n+1} = s_{n+1}$. Since v_{n+1} can only take values 1 and -1, the only possible values of $\overline{v}_{n+1}$ are -1 and 1. The only possibility for it to take the value 1 is when $v_{n+1} = 1$ for some $(1-\varepsilon)$-solution. In this case, from the last equation, we conclude that $s_1 \cdot v_1 + \ldots + s_n \cdot v_n = 0$, i.e., that the values $v_j \in \{-1, 1\}$ form a solution to the original instance of the *PARTITION* problem. The reduction is complete and thus, the problem of computing $\overline{v}_j$ is also NP-hard. The proposition is proven.

Comment. A similar NP-hardness result, for linear systems over the finite field $Z/2Z$, was recently proven in [4] (see also [15]).

Proof of Proposition 2. Tarski-Seidenberg algorithm [14, 12, 7] handles formulas from the language which is very similar to the one we have described, but with no variables for intervals; this language (we will denote it by $L_{\rm real}$) is called *first order theory of real numbers*. Namely, this algorithm does the following:

- For any formula from $L_{\rm real}$ *without* any free variables, this algorithm checks whether the given formula is true or not.
- For any formula *with* free variables $x_1, \ldots, x_n$, the algorithm generates an equivalent formula without quantifiers, i.e., a formula which is obtained from elementary formulas of the type $P(x_1, \ldots, x_n) = 0$, $Q(x_1, \ldots, x_n) \geq 0$, $R(x_1, \ldots, x_n) > 0$, with polynomial P, Q, R, ..., by logical connectives $\vee$, $\&$, $\neg$. (The main algorithmic advantage of this equivalent representation is that for any given rational

numbers $x_1, \ldots, x_n$, it was not clear how to check whether the original formula was true, but checking the new formula is straightforward.)

We want to apply this algorithm to our case as well. For this, we will show that each formula from $L_{\rm int}$ can be reformulated as an equivalent formula from the language $L_{\rm real}$. Since $L_{\rm int}$ is obtained from $L_{\rm real}$ by adding interval variables, we must, therefore, for this reformulation to be successful, somehow "get rid" of interval variables. This is rather easy to do:

- each interval variable $\mathbf{x} = [\underline{\mathbf{x}}, \overline{\mathbf{x}}]$ can be represented as a pair of real variables $\underline{x}$, $\overline{x}$ with an additional condition $\underline{x} \le \overline{x}$; and
- each elementary formula of the type $t \in \mathbf{x}$ can be reformulated as $\underline{x} \le t \,\&\, t \le \overline{x}$.

The reduction is complete, and thus, the proposition is proven.

Proof of Proposition 3. In this proof, we will use the same Tarski-Seidenberg theorem which we used in the proof of Proposition 2. We want to describe the set of all solutions to a multi-interval linear system. The only part of the definition of this solution which is not already in the language $L_{\rm real}$ is the formula $y_k \in \mathbf{y}_k$ for a multi-interval $\mathbf{y}_k$. Therefore, if we want to describe the notion of a solution in $L_{\rm real}$, we must describe this formula in $L_{\rm real}$.

By definition, a multi-interval is a finite union of generalized intervals: $\mathbf{y}_k = S_1 \cup \ldots \cup S_p$. Thus, the formula $y_k \in \mathbf{y}_k = S_1 \cup \ldots S_p$ can be reformulated as $y_k \in S_1 \vee \ldots \vee y_k \in S_p$. For each generalized interval S_q, we can easily reformulate the formula $y_k \in S_q$ in terms of $L_{\rm real}$: e.g., $y_k \in (a, b)$ is equivalent to $a < y_k \,\&\, y_k < b$; $y_k \in (a, \infty)$ is equivalent to $a < y_k$, etc. Thus, the condition that $x_1, \ldots, x_n$ form a solution of a multi-interval algebraic system can be reformulated in $L_{\rm real}$. Thus, consistency of a system, i.e., the fact that $\exists x_1 \ldots \exists x_n$ for which $(x_1, \ldots, x_n)$ form a solution, is also equivalent to a formula from $L_{\rm real}$, and Tarski's algorithm can decide whether the resulting formula is true or not (and thus, whether the original system was consistent).

Similarly, the condition that $x_j \in \mathbf{x}_j$ (i.e., that $\exists x_1 \ldots \exists x_{j-1} \exists x_{j+1} \exists x_n$ for which $(x_1, \ldots, x_{j-1}, x_j, x_{j+1}, \ldots, x_n)$ is a solution) can also reformulated in $L_{\rm real}$. Thus, if we apply Tarski-Seidenberg algorithm to the resulting formula, we will get a quantifier-free equivalent formula that describes the same condition $x_j \in \mathbf{x}_j$, i.e., a formula which is a logical combination of elementary formulas of the type $P(x) = 0$, $Q(x) \ge 0$, and $R(x) > 0$, where $P(x)$, $Q(x)$, and $R(x)$ are polynomials with rational coefficients. For each such polynomial, we can compute the roots, and each condition can be expressed as x belonging to a finite union of generalized intervals with these roots as endpoints. Thus, each of these conditions defines a multi-interval and therefore, their logical combination also defines a multi-interval, with computable endpoints. The proposition is proven.

Proof of Proposition 4. In this proof, we will use the result of Matiyasevich *et al.* [10, 11, 2] that no algorithm is possible to solve Diophantine equations with 13 variable, i.e., no algorithm can decide whether a formula

$$\exists x_1 \ldots \exists x_{13} [(x_1 \in N) \,\&\, \ldots \,\&\, (x_{13} \in N) \,\&\, Q(x_1, \ldots, x_{13}) = 0]$$

is true, where N denotes the set of all natural numbers and Q is a polynomial with integer coefficients. (This result solved tenth Hilbert problem [5].)

We want to re-formulate the above formula as an equivalent muti-interval formula. The only part that needs to be reformulated is the part $x \in N$. Let us show that this formula is equivalent to the following formula from $L_{\rm mult}$: $\forall \mathbf{x}\, [P(\mathbf{x}) \to x \in \mathbf{x}]$, where by $P(\mathbf{x})$, we denoted the following formula:

$$0 \in \mathbf{x} \,\&\, \forall y\, (y \in \mathbf{x} \to \exists z\, (z = y + 1 \,\&\, z \in \mathbf{z})).$$

Let us show that these formulas are indeed equivalent.

First, let us assume that $x \in N$. Then, if the multi-interval $\mathbf{x}$ satisfies the property $P(\mathbf{x})$, this means that it contains 0, and with every element y, it also contains $z = y + 1$. By induction, we can conclude that $\mathbf{x}$ contains all natural numbers, and therefore, that $x \in \mathbf{x}$.

Second, let us assume that x satisfies the above property from $L_{\rm mult}$. Let us then prove that x is a natural number. Indeed, let us denote $n = \lceil |x| \rceil + 1$; then, n is a natural number for which $x < n$. Let us now take the following multi-interval: $\mathbf{x} = [0, 0] \cup [1, 1] \cup \ldots \cup [n - 1, n - 1] \cup [n, \infty)$. It is easy to check that $\mathbf{x}$ satisfies the property $P(\mathbf{x})$, and therefore, we can conclude that $x \in \mathbf{x}$. We know that $x < n$, and by the definition of $\mathbf{x}$, the only elements from $\mathbf{x}$ which are smaller than n are natural numbers $0, 1, \ldots, n - 1$. Hence, x is a natural number.

Since the formula $x_i \in N$ can be reformulated in $L_{\rm mult}$, we can thus reformulate the original Matiyasevich's formula in this language. Therefore, the problem of checking whether a given formula from $L_{\rm mult}$ is true is not algorithmically solvable: because if it was, we could apply the algorithm to translations of Matiyasevich formulas, which constradicts to Matiyasevich's result.

To complete the proof, let us show that checking consistency and computing solutions are also algorithmically un-solvable tasks.

- For any formula F without free variables, we can form a generalized multi-interval condition $F \,\&\, x_1 = 1$. This condition is consistent if and only if the formula F is true; since it is impossible to check whether a formula is true, it is also impossible to check whether a given condition is consistent.
- Similarly, for any formula F without free variables, we can form a generalized condition $(F \,\&\, x_1 = 1) \vee (\neg F \,\&\, x_1 = 0)$. This condition is consistent, because no matter whether F is true or not, we have a solution (either $x_1 = 1$ or $x_1 = 0$). However, if we could have an algorithm for producing a solution, we would then be able to tell whether the formula F is true or not, and we already know that this is impossible. Thus, no algorithm can always compute a solution.

The proposition is proven.

Comment. If, instead of allowing multi-intervals with arbitrary number of components, we set an upper bound B on the number of components, then we can express each formula $x \in \mathbf{x}$ in terms of $L_{\rm real}$ (as we did in the proofs of Propositions 2 and 3), and hence, both problems (of checking consistency and of computing the solution) become algorithmically solvable.

Proof of Proposition 5. For *discrete* time, algorithmical solvability follows from the applicability of Tarski-Seidenberg algorithm, because in discrete-time case, we have finitely many $(n \cdot (T + 1 - T_0))$ variables $x_i(t)$, $1 \le i \le n$, $T_0 \le i \le T$, and the relation between these variables (dynamical and consistency at T_0 and T) can be easily reformulated in the language $L_{\rm real}$.

Let us show that for *continuous* time, the problem is not algorithmically solvable. We will show that it is not solvable even for the simplest case $T_0 = 0$ and $T = 1$. For this proof, we will use the same Matiyasevich's result as in the proof of Proposition 4. According to this result, no algorithm can tell whether a given polynomial equation $Q(n_1, \ldots, n_{13}) = 0$ (with integer coefficients) has a solution in which all the values n_i are natural numbers. It is also known (see, e.g., [2]) that a similar negative result holds if we are looking for *integer* solutions (not necessarily non-negative integer). Indeed, it is known that each natural number can be represented as a sum of four squares of integers: $n_i = v_{i,1}^2 + \ldots + v_{i,4}^2$. Thus, the equation $Q(n_1, \ldots, n_{13}) = 0$ has a natural-number solution if and only if the equation $R(v_{1,1}, \ldots, v_{13,4}) = 0$ has an integer solution, where $R = Q(v_{1,1}^2 + \ldots + v_{1,4}^2, \ldots, v_{13,1}^2 + \ldots + v_{13,4}^2)$.

Let us show that for every polynomial $R(v_1, \ldots, v_m)$ with integer coefficients, the existence of an integer solution can be reduced to solving an appropriate system of differential equations. This new system will have:

- $n = 3m + 3$ variables $v_1, \ldots, v_m$, v_0, p, s, c, s_i and c_i $(1 \le i \le m)$;
- the following equations: $\dot{v}_i = 0$, $\dot{v}_0 = R(v_1, \ldots, v_m)$, $\dot{p} = 0$, $\dot{s} = p \cdot c$, $\dot{c} = -p \cdot s$, $\dot{s}_i = v_i \cdot p \cdot c_i$, and $\dot{c}_i = -v_i \cdot p \cdot s_i$;
- initial interval state, in which $\mathbf{v}_1 = \ldots = \mathbf{v}_m = (-\infty, \infty)$, $\mathbf{v}_0 = [0, 0]$, $\mathbf{p} = [3, 4]$, $\mathbf{s} = \mathbf{s}_1 = \ldots = \mathbf{s}_m = [0, 0]$, and $\mathbf{c} = \mathbf{c}_1 = \ldots = \mathbf{c}_m = [1, 1]$;
- final interval state, in which $\mathbf{v}_1 = \ldots = \mathbf{v}_m = (-\infty, \infty)$, $\mathbf{v}_0 = [0, 0]$, $\mathbf{p} = [3, 4]$, $\mathbf{s} = \mathbf{s}_1 = \ldots = \mathbf{s}_m = [0, 0]$, and $\mathbf{c} = \mathbf{c}_1 = \ldots = \mathbf{c}_m = (-\infty, \infty)$.

Let us show that this problem has a solution if and only if the equation $R(v_1, \ldots, v_m) = 0$ has an integer solution.

Indeed, if the equation $R = 0$ has an integer solution $v_1, \ldots, v_m$, then we can take $v_i(t) = v_i$, $v_0(t) = 0$, $p(t) = \pi$, $s(t) = \sin(\pi \cdot t)$, $c(t) = \cos(\pi \cdot t)$, $s_i(t) = \sin(\pi \cdot v_i \cdot t)$, and $c_i(t) = \cos(\pi \cdot v_i \cdot t)$. One can easily check that this state is indeed a solution to the above system of differential equations, and that the states $s(T_0$ and $s(T)$ are consistent with the given interval states.

Vice versa, let us assume that the problem of solving a differentuial equation has a solution. Since $\dot{v}_i = 0$, the values $v_i(t)$ do not change in time. Let us show that these values satisfy the equation $R(v_1, \ldots, v_m) = 0$, and that they are integers. Indeed, since $v_i = {\rm const}$, we have $R(v_1, \ldots, v_m) = {\rm const}$, and therefore, $v_0(t) = v_0(0) + t \cdot R(v_1, \ldots, v_m)$ for all t; in particular, $v_0(1) = v_0(0) + R(v_1, \ldots, v_m)$. From the consistency with the

interval states, we know that $v_0(0) = v_0(1) = 0$, therefore, we can conclude that $R(v_1, \ldots, v_m) = 0$. So, to complete our proof, it suffices to show that all the values v_i are integers.

To prove this, let us first prove that $p = \pi$. Indeed, from $\dot{p} = 0$, we conclude that p is a constant. Now, from the equations $\dot{s} = p \cdot c$ and $\dot{c} = -p \cdot s$, we conclude that both $s(t)$ and $c(t)$ are linear combinations of the functions $\sin(p \cdot t)$ and $\cos(p \cdot t)$. From the initial conditions $s(0) = 0$ and $c(0) = 1$, we conclude that $s(t) = \sin(p \cdot t)$ and $c(t) = \cos(p \cdot t)$. Now, from the consistency with the final condition $\mathbf{s}(1) = [0, 0]$, we conclude that $\sin(p) = 0$, i.e., that $p = k \cdot \pi$ for some inetger k. Since we know that $p \in \mathbf{p} = [3, 4]$, the only possibility is $k = 1$, i.e., $p = \pi$.

Similarlly, from the facts that p and v_i are constants, and from the differential equations $\dot{s}_i = v_i \cdot p \cdot c_i$ and $\dot{c}_i = -v_i \cdot p \cdot s_i$ and the initial conditions $s_i(0) = 0$ and $c_i(0) = 1$, we conclude that $s_i(t) = \sin(v_i \cdot p \cdot t)$ and $c_i = \cos(v_i \cdot p \cdot t)$. Thus, from the consistentcy with the final state, we conclude that $s_i(1) = 0$ and therefore, that $\sin(v_i \cdot p) = \sin(v_i \cdot \pi) = 0$. This means that v_i is an integer. The reduction is proven, and so is the proposition.

Acknowledgments. This work was supported in part by NASA under cooperative agreement NCC5-209, by NSF grant No. DUE-9750858, by United Space Alliance, grant No. NAS 9-20000 (P.O. 297A001153), by the National Security Agency, and by the Future Aerospace Science and Technology Program (FAST) Center for Structural Integrity of Aerospace Systems, effort sponsored by the Air Force Office of Scientific Research, Air Force Materiel Command, USAF, under grant number F49620-95-1-0518.

The author is thankful to Luc Longpré and to all participants of the International Conference Interval'98 for valuable comments.

References

[1] Th. H. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to algorithms*, MIT Press, Cambridge, MA, and Mc-Graw Hill Co., N.Y., 1990.

[2] M. Davis, Yu. V. Matiyasevich, and J. Robinson, "Hilbert's tenth problem. Diophantine equations: positive aspects of a negative solution", In: *Mathematical developments arising from Hilbert's problems*, Proceedings of Symposia in Pure Mathematics, Vol. 28, American Math. Society, Providence, RI, 1976, Part 2, pp. 323–378.

[3] M. E. Garey and D. S. Johnson, *Computers and intractability: a guide to the theory of NP-completeness*, Freeman, San Francisco, 1979.

[4] J. Hastad, "Some optimal inapproximability results", *Proceedings of the Annual ACM Symposium on Theory of Computing STOC'97, El Paso, TX, May 4–6, 1997*, ACM Press, 1997, pp. 1–10.

[5] D. Hilbert, "Mathematical Problems" (lecture delivered before the International Congress of Mathematics in Paris in 1900), translated in *Bull. Amer. Math, Soc.*, 1902, Vol. 8, pp. 437–479; reprinted in *Mathematical developments arising from Hilbert's problems*, Proceedings of Symposia in Pure Mathematics, Vol. 28, American Math. Society, Providence, RI, 1976, Part 1, pp. 1–34.

[6] V. Kreinovich, A. V. Lakeyev, and S. I. Noskov, "Optimal solution of interval linear systems is intractable (NP-hard)." *Interval Computations,* 1993, No. 1, pp. 6–14.

[7] V. Kreinovich, A. Lakeyev, J. Rohn, and P. Kahl, *Computational complexity and feasibility of data processing and interval computations*, Kluwer, Dordrecht, 1997.

[8] A. V. Lakeyev and S. I. Noskov, "A description of the set of solutions of a linear equation with interval defined operator and right-hand side" *Russian Academy of Sciences, Doklady, Mathematics*, 1993, Vol. 47, No. 3, pp. 518–523.

[9] A. V. Lakeyev and S. I. Noskov, "On the solution set of a linear equation with the right-hand side and operator given by intervals", *Siberian Math. J.*, 1994, Vol. 35, No. 5, pp. 957-966.

[10] Yu. V. Matiyasevich, "Enumerable sets are diophantine", *Soviet Math. Doklady*, 1970, Vol. 11, pp. 354–357.

[11] Yu. V. Matiyasevich and J. Robinson, "Reduction of an arbitrary Diophantine equation to one in 13 unknowns", *Acta Arithmetica*, 1974, Vol. 27, pp. 521–553.

[12] A. Seidenberg, "A new decision method for elementary algebra", *Annals of Math.*, 1954, Vol. 60, pp. 365–374.

[13] S. P. Shary, "Algebraic approach to the interval linear static identification, tolerance, and control problems, or One more application of Kaucher arithmetic", *Reliable Computing*, 1996, Vol. 2, No. 1, pp. 3–34.

[14] A. Tarski, *A decision method for elementary algebra and geometry*, 2nd ed., Berkeley and Los Angeles, 1951.

[15] U. Zwick, "Finding almost-satisfying assignments", *Proceedings of the Annual ACM Symposium on Theory of Computing STOC'98, Dallas, TX, May 23–26, 1998*, ACM Press, 1998, pp. 551–560.