Interval Computations
No 2, 1993

UniCalc, a Novel Approach to Solving Systems of Algebraic Equations

A. B. Babichev, O. B. Kadyrova, T. P. Kashevarova,
A. S. Leshchenko, and A. L. Semenov

This paper describes a novel approach to solving systems of algebraic equations and inequalities that is based on *subdefinite calculations*. The use of these methods makes it possible to solve overdetermined and underdetermined systems, as well as systems with imprecise and incomplete data. The approach was implemented with the help of the methods of *interval mathematics*. *The UniCalc solver*, also described in this paper, was developed on the basis of this approach. To illustrate the capabilities of UniCalc, we give examples of problems solved with its help.

UniCalc: новый подход к решению систем алгебраических уравнений

А. Б. Бабичев, О. Б. Кадырова, Т. П. Кашеварова,
А. С. Лещенко, А. Л. Семенов

Описан новый подход к решению систем алгебраических уравнений и неравенств. Данный подход основывается на аппарате *недоопределенных вычислений* и позволяет решать недоопределенные и переопределенные системы уравнений, а также задачи с неточными и неполными данными. Для реализации аппарата недоопределенных вычислений используются методы *интервальной математики*. В статье также описывается *решатель UniCalc*, реализованный на базе рассматриваемого аппарата. Для иллюстрации возможностей UniCalc'а приводятся примеры задач, решенных с его помощью.

1 Introduction

Solving systems of algebraic equations attracts considerable interest, and numerous methods exist for solving such systems. As a rule, these algorithms are based on approximate methods of numerical mathematics; in the following they are referred to as the classical methods. (Examples of such methods are Newton's method, bisection methods, relaxation methods, gradient methods etc.) Normally, each classical method is designed for a particular class of systems of equations. Moreover, most of these methods are iterative and require a good initial approximation. In addition, the classical methods are good only for systems that are fully defined with deterministic equations. However, the systems that describe phenomena of real life may be either underdetermined or overdetermined; they may include inequalities, uncertain and imprecise values. To solve such classes of problems, several approaches exist.

One is *interval analysis* [1, 2] allowing some classical methods to be adapted to systems with imprecise data and making it possible to tackle some problems unsuitable for classical methods.

There are some other approaches to problems with uncertain data such as *fuzzy set theory* [3]. The degree of uncertainty is described by some function called the membership function.

To solve real systems containing additional conditions formulated as equations and inequalities, an approach called *constraint propagation* is used [4]. Although this method makes it possible to consider certain constraints, it fails in the same manner as classical methods with underdetermined and overdetermined systems with imprecise data. One way to obviate these drawbacks is to combine constraint propagation methods with interval mathematics [5].

This paper considers an approach for solving algebraic equations created in the framework of artificial intelligence research [6, 7]; this approach is free of almost all drawbacks of the aforementioned approaches. It is applicable to a wide class of systems and does not require any initial approximations. This approach is called *the method of subdefinite calculations*. It has some parallels with the constraint propagation method, except for its greater generality. Based on the method of subdefinite calculations, the UniCalc solver was developed in the Novosibirsk Division of the Russian Research Institute

of Artificial Intelligence.

The rest of the paper is organized as follows. Sections 2 and 3 present the algorithm of subdefinite calculations and some implementation questions. Sections 4, 5, and 6 consider both the UniCalc solver as a whole and some of its components. Numerical experiments with UniCalc are described in Section 7. Section 8 provides a summary, UniCalc's technical characteristics and some conclusions.

2 The algorithm

This section briefly describes the algorithm used by the UniCalc solver. The algorithm is fully described in [6, 7].

2.1 Concepts and designations

Let a calculation model M be determined by the set of variables X and the set of relations over these variables R. We shall denote the model by $M = (X, R)$. Let A be the value domain of the model's variables, and let ${}^*A$ denote the set of all nonempty subsets of the set A. The values ${}^*a \in {}^*A$ containing one member only are called *precise*, while other values are called *subdefinite*. The value ${}^*a$ corresponding to the entire set A is *the fully indefinite value*. Let us map uniquely each variable x in the set X into a variable ${}^*x$ whose value domain is the set ${}^*A$. This maps the set of variables X onto a set ${}^*X$ of variables ${}^*x$. The variables ${}^*x$ will be called *subdefinite variables*.

The subdefinite description of the model M is a set (M, h), where $h = ({}^*a_1, \ldots, {}^*a_n)$ is a vector of subdefinite values. Narin'yani showed in [6, 7] that for subdefinite descriptions it is possible to construct a finite automaton that generates finite sequences of states. The sequences are terminated either with the 'end' state, or the 'conflict' state. In the first case, the automaton produces a vector h which is an n-dimensional parallelepiped in the space A^n containing the set of the values of the variables X that satisfy the model M.

The process of inference/calculations over models usually involves interpreting each relation with a help of a set of functions allowing us to find the values of other variables from known values of some variables. For example, the relation

$$e^a - 2b^2 + c + 7 = 0$$

is interpreted via the following three functions:

$$a := \ln(2b^2 - c - 7); \quad b := \left((e^a + c + 7) : 2\right)^{\frac{1}{2}}; \quad c := 2b^2 - e^a - 7.$$

Since the variables ${}^*X$ rather than X are used while working with subdefinite descriptions, it is necessary to define operations over subdefinite values and subdefinite functions. Narin'yani showed [6] that if we map each m-ary operation s over values from A into an operation ${}^*s$ over undefined values from ${}^*A$ according to the formula

$${}^*s({}^*a_1, \ldots, {}^*a_m) = \left\{a = s(b) \mid b \in {}^*a_1 \times {}^*a_2 \times \cdots \times {}^*a_m\right\} \qquad (1)$$

and if each m-ary function f of variables in X is mapped into a function ${}^*f$ of variables in ${}^*X$ by the formula

$${}^*f({}^*x_1, \ldots, {}^*x_m) = \left\{x = f(b) \mid b \in {}^*x_1 \times {}^*x_2 \times \cdots \times {}^*x_m \cap R\right\} \qquad (2)$$

where R is an additional relation connecting the variables $x_1, \ldots, x_m$ (for uncorrelated variables $R = {}^*A^m$), then it is possible to use the inference/calculation apparatus for the initial models for inference/calculations on subdefinite models.

2.2 The calculation algorithm for subdefinite models

Suppose we have a model with a subdefinite description (in [7] such models are called *GCM, generalized calculation models*). According to Section 2.1, the inference/calculation procedure for such models may be described as a procedure of calculating the interpretation functions of subdefinite variables. If a certain relation of the initial model relating variables $x_1, \ldots, x_m$ is interpreted by a set of the following functions:

$$x_i := f_i(x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_m), \qquad i = 1, \ldots, m \qquad (3)$$

then, using formula 2, the corresponding interpretation functions for the subdefinite model may be defined as follows:

$$ {}^*x_i := {}^*f_i({}^*x_1, \ldots, {}^*x_{i-1}, {}^*x_{i+1}, \ldots, {}^*x_m) \cap {}^*a_i, \qquad i = 1, \ldots, m \quad (4) $$

where ${}^*a_i$ is the current value of ${}^*x_i$.

The inference/calculation process formulated in [7] for GCM may be presented in the form of the following algorithm (let ${}^*X_t$ denote the set of the variables in ${}^*X$ whose values have become more precise at the step t, and let $F \mid {}^*X_t$ denote the set of interpretation functions having at least one argument which belongs to ${}^*X_t$):

Step 1. $t = 1$; ${}^*X_t = \{{}^*x_1, \ldots, {}^*x_n\}$.

Note that all model variables have values: they are either initialized or their value is fully indefinite, i.e. the set A.

All the model interpretation functions are placed in set $F \mid {}^*X_1$ and all the members of this set form the set of active functions.

Step 2. Combine the function set $F \mid {}^*X_t$ with the active interpretation functions.

Step 3. Choose an arbitrary subset in the produced set. This subset is joined to the working function set and removed from the active function set.

Step 4. Remove an arbitrary subset of functions from the working function set. The functions of the subset are calculated at this step. The results of these calculations are compared with the values of subdefinite variables in the common memory, and variables that have changed their values form the set ${}^*X_{t+1}$ defining set the $F \mid {}^*X_{t+1}$.

If a subdefinite variable yields, as a result of calculating some interpretation function, the value that is equal to the empty set the model is incompatible. The algorithm terminates.

Step 5. If the set of active functions, the working function set and the set ${}^*X_{t+1}$ are simultaneously empty, go to Step 7.

Step 6. $t = t + 1$ and go to Step 2.

Step 7. Output the values in the set ${}^*X = \{{}^*x_1, \ldots, {}^*x_n\}$.

This algorithm has the following features:

- The model can be underdetermined or overdetermined and the parameters of the model may be imprecise or unknown;
- No distinction is made between the model's arguments and results: all variables have values that are subdefinite in various degrees; the calculations are performed for all variables of the model;
- This algorithm determines a parallel, asynchronous, undetermined process with data-driven flow control;
- According to formula 4, subdefiniteness of the variables never increases and the calculation is converging. For all models having only finite subdefinite values, this procedure terminates in a finite number of steps.

Consider an example. Let the model be specified by two equations:

$$8a - b = 15, \qquad 6b + 4c = 230$$

and suppose that it is known that a, b, and c are integers in the range from 1 to 100. Hence, the subdefinite value of each of these variables can be represented by a set of integers with the initial state $A = \{1, \ldots, 100\}$. These relations are interpreted by the following functions:

$$\begin{aligned} f_1 &: a := (b + 15)/8 \\ f_2 &: b := 8a - 15 \\ f_3 &: b := (230 - 4c)/6 \\ f_4 &: c := (230 - 6b)/4. \end{aligned}$$

Introducing the subdefinite description, we have:

$$\begin{aligned} {}^*f_1 &: {}^*a := \left\{a = (b + 15)/8 \mid b \in {}^*b\right\} \cap {}^*a \\ {}^*f_2 &: {}^*b := \left\{b = 8a - 15 \mid a \in {}^*a\right\} \cap {}^*b \\ {}^*f_3 &: {}^*b := \left\{b = (230 - 4c)/6 \mid c \in {}^*c\right\} \cap {}^*b \\ {}^*f_4 &: {}^*c := \left\{c = (230 - 6b)/4 \mid b \in {}^*b\right\} \cap {}^*c. \end{aligned}$$

In compliance with the above algorithm, we obtain

$${}^*X_1 = \left\{{}^*a = \{1, \ldots, 100\}, {}^*b = \{1, \ldots, 100\}, {}^*c = \{1, \ldots, 100\}\right\}$$
$$F \mid {}^*X_1 = \left\{{}^*f_1, {}^*f_2, {}^*f_3, {}^*f_4\right\}.$$

The table below illustrates the algorithm's execution:

Subdefinite values	**Working function**	**Active functions**
${}^*a = \{2, 3, \ldots, 14\}$	${}^*f_1$	${}^*f_2$, ${}^*f_3$, ${}^*f_4$
${}^*b = \{1, 9, \ldots, 97\}$	${}^*f_2$	${}^*f_3$, ${}^*f_4$
${}^*b = \{1, 9, 17, 25, 33\}$	${}^*f_3$	${}^*f_1$, ${}^*f_4$
${}^*c = \{8, 20, 32, 44, 56\}$	${}^*f_4$	${}^*f_1$, ${}^*f_3$
${}^*a = \{2, 3, 4, 5, 6\}$	${}^*f_1$	${}^*f_2$, ${}^*f_3$
${}^*b = \{1, 9, 17, 25, 33\}$	${}^*f_3$	${}^*f_2$
${}^*b = \{1, 9, 17, 25, 33\}$	${}^*f_2$	

Therefore, the algorithm yields the following result:

$$\big\{{}^*a = \{2,3,4,5,6\}, \quad {}^*b = \{1,9,17,25,33\}, \quad {}^*c = \{8,20,32,44,56\}\big\}.$$

To find an every separate solution one should combine the values from different sets and verify the system. In our case such solutions are:

$$\{2,1,56\}, \quad \{3,9,44\}, \quad \{4,17,32\}, \quad \{5,25,20\}, \quad \{6,33,8\}.$$

3 Implementation of the algorithm

The implementation of the algorithm just described assumes the choice of a set (sets) A corresponding to the class of models; in addition acceptable operations and functions over values in A and variables in X are to be defined and extended for objects in ${}^*A$ and ${}^*X$. Furthermore, the model should be represented in a form ensuring efficient organization of the calculation process. To represent subdefinite objects, the active data type apparatus [8] has been proposed and used to implement various types of subdefinite data (integers and reals, sets, logical and enumerable data, objects of planimetry, etc.) [9].

3.1 Implementation of subdefinite numbers

We consider only computational models over real numbers; therefore the set A is the field of real numbers R or the ring of integer numbers Z. We assume that all the arithmetic operations $\{+, -, *, /\}$ as well as exponentiation

and root extraction are defined over values in the set A. The set of acceptable functions includes exponential, logarithmic and all direct trigonometric functions. Since computer number representations are finite, the set A will be represented by finite intervals in R or Z which are bounded from below and from above by some boundary numbers Min$A$ and Max$A$ playing the role of $-\infty$ and $+\infty$, respectively (these numbers differ for real and integer sets A). The operations on elements of A are defined as follows: if $a, b \in A$, and $\odot$ is some operator, then

$$\begin{aligned} a \odot b &= \max(a \odot b, \mathrm{Min}A) \qquad \text{if } a \odot b < 0 \\ a \odot b &= \min(a \odot b, \mathrm{Max}A) \qquad \text{if } a \odot b \geq 0. \end{aligned}$$

Assuming these conventions and considering ${}^*A$ as the set of all intervals in the set A, we can extend all operations on elements of A to operations on ${}^*A$ defined by formula 1, using the appropriate operations of interval mathematics. Note that any nonempty subset of the set A may be embedded into some interval. Therefore, assuming that ${}^*A$ is the set of all intervals of A, we can only increase subdefiniteness of solutions without losing any of them. These operations must be supplemented by the operations of conversion of subdefinite types and of assigning subdefinite values. Alefeld [2] showed that interval continuous analytic functions are monotonic by inclusion, i.e. if f is a continuous analytic function of the interval variables $X_1, \ldots, X_n$ and $X_1 \subseteq Y_1, \ldots, X_n \subseteq Y_n$, then $f(X_1, \ldots, X_n) \subseteq f(Y_1, \ldots, Y_n)$. In view of this property, to extend functions over variables from A, it is possible to use their interval extensions in the corresponding continuity domains, since the interval thus obtained will always include the set determined by formula 2. Considering the possibility of using intervals with infinite limits, we can continue interval expansion to discontinuous functions, assuming, for example, that $1/0 = (0, \mathrm{Max}A)$ and $1/(-1, 1) = (\mathrm{Min}A, \mathrm{Max}A)$. Such an approach makes it possible to avoid the divide-by-zero situation that is fatal for other methods.

3.2 Some implementation problems

Beside the aforementioned divide-by-zero problem, some other implementation problems have been solved. They are mostly due to violation of uniqueness in computing some functions of interval parameters (for instance, for

the inverse trigonometric functions or for even power roots) which may require using of multiintervals (unavailable in the current implementation). The violation of uniqueness in the course of calculations is rather a common problem since for each relation we have to calculate the interpretation functions for each of its variables. To overcome this difficulty, the intervals of function monotonicity are extracted to calculate the function's values, and the general solution is constructed as the union of all solution intervals obtained. Suppose we have the model presented below.

$$x^2 - a = 0$$
$$4 \leq a \leq 25$$

The interpretation function $x := \sqrt{a}$ yields two intervals $x_1 = (-5, -2)$ and $x_2 = (2, 5)$, and the interval $x = (-5, 5)$ will be returned as the result. However, if the relation $x > 0$ is added to the system, the resulting interval will be $x = (2, 5)$.

Another problem of interval calculations worth mentioning is the problem of indefiniteness of the origin. Indefiniteness arises, for instance, when evaluating expressions like x/y or x^y, where x and y are intervals containing zero. The method for handling such situations is chosen by the user who can use options to specify whether the presence of such arguments in these expressions is contradictory or not. The latter case requires the user to specify what should be considered to be the result of the expression — the fully indefinite value or the value defined by continuity.

3.3 Representation of models

The most appropriate method for the algorithm described above is the network representation of models. The virtual flow data-driven processor [10, 11] is based on such a representation using the apparatus of active data types. It is the kernel of the UniCalc system. In this processor, a network is represented as a bipartite oriented graph with two types of vertices: objects and operators. Objects represent model variables, and operators represent functional links between objects (interpretation functions). The outgoing edges point to the operators whose arguments are the corresponding objects, and incoming edges specify objects which provide values to these operators. A network is associated with a discipline of its execution, which

substantiates the inference algorithm. This substantiation depends on the computer's architecture, its computational capabilities, as well as a number of other factors. For instance, sets may be selected at Step 3 according to certain priorities, from a queue or randomly. Interpretation functions are computed at Step 5 sequentially or in parallel, etc. In the current implementation of the processor, the interpretation functions are selected depending on their ordinal number and are computed sequentially. If one uses computers designed for parallel processing or transputers, the strategy of network execution should be changed.

4 The UniCalc solver

The algorithm described above provides a novel approach to solving systems of algebraic equations that describe the class of models under consideration. To use it efficiently for practical purposes, we have developed the UniCalc solver whose nucleus is the flow processor considered in Section 3.

UniCalc is an abbreviation for Universal Calculator, indicating the ability of the system to solve a very broad class of mathematical problems.

4.1 Purpose and capabilities

The UniCalc solver was designed to solve arbitrary systems of algebraic and algebraic-differential relations. For this program, a relation is considered to be an equation, inequality or a logical expression. According to the algorithm used, the system to be solved can be either overdetermined or underdetermined, and the system's parameters (coefficients, variables, initial conditions for the Cauchy problem) can be subdefinite and expressed as intervals. Such a system may contain only integer and real variables or combine both integer and real variables.

As a result of solving algebraic systems, we either find a parallelepiped that contains all roots of the system, or a message about the system's incompatibility is issued. If the system has a single root, then the parallelepiped will in most cases be reduced to a point (with a given accuracy). If the system has several roots, to locate each of them it is necessary to add the appropriate relations, or use the built-in tool for automatic root locating.

4.2 Architecture of the UniCalc solver

The solver is an integrated environment supporting input of the system to be solved, its modification, calculations, viewing results, specifying accuracy, etc. To input and modify the system UniCalc has the built-in text editor. To write the problems, a source language close to commonly used mathematical notation is provided. All problems are processed with the above algorithm. To translate the source input into a network, the solver includes a translator and pre-processors, in particular, for symbolic transformations and solution of systems of algebraic-differential relations. The first pre-processor simplifies the system, differentiates symbolically and makes transformations needed to solve linear systems. The second pre-processor translates the system from the differential equation language into the basic UniCalc source language that supports solving problems of this kind.

The UniCalc's user interface offers a number of services to support problem solving, including various setup options. When solving problems requiring large computation times, it is possible to suspend calculations to see the intermediate results, and depending on convergence rate, either to continue computation or to stop it. A feature for locating roots is used when exact solutions are in large intervals. This process involves dichotomic division of the obtained interval for the selected variable. This tool is useful to find global function extreme values from the right to left or the left to right along the function value interval.

5 Symbolic pre-processor

The main objective of this pre-processor is optimization of the network and performing some types of symbolic manipulations. The system to be solved by UniCalc is represented as a functional network whose size depends upon the number of variables and expressions in the system. The pre-processor translates the source expressions into its internal representation (which is a dynamic modification of n-ary Kantorovich schemata), simplifying the source expressions and storing only one copy of each subexpression. Note that reducing similar terms optimizes networks and sometimes narrows down intervals. For example, the following expressions $x \times x$ and $x = (-4, 5)$, if evaluated directly, yield $(-20, 25)$, whereas the symbolic pre-processor transforms these expressions to x^2, resulting in the interval $(0, 25)$. In addi-

tion, storing only one copy of each term reduces the number of intermediate calculations and reduces subdefiniteness of results, as well as computation time.

Symbolic calculations are performed with the internal representation. Currently these calculations cover only differentiation. Expressions obtained in this way make up the system used to create the network used for calculations. Such an approach not only offers symbolic transformations, but helps to solve the problem of a symbolic-numerical interface. Prior to constructing a network it is possible to view, print out, or store the expressions obtained as a result of reductions and symbolic transformation.

Here, we notice the usefulness of symbolic differentiation to study behavior of functions and to simplify the statement of problems, as well as to solve optimization problems. UniCalc does not support solving such problems explicitly, but for continuous real variable functions this problem can be stated as follows. If the first derivative at an extremum is zero, the second derivative sign at this point determines whether it is a maximum or a minimum (see the example in Section 7). If the function has more than one extremum of this type, then an interval containing all these points will be returned. To separate individual extreme points, the automatic root locating procedure with an appropriate selection of the search direction may be used.

In addition, the symbolic pre-processor incorporates some features eliminating the drawbacks exposed in using UniCalc, one of which is low speed on systems of linear algebraic equations. To eliminate this, symbolic variable substitution is performed to obtain a triangular (trapezoid in general case) system of equations where some variables are expressed via others. If the system is subdefinite, an interval solution will be obtained.

6 Differential equations

UniCalc can solve systems of algebraic-differential equations supported by the differential equation pre-processor [12]. The system of equations applied to the input of this pre-processor is a mathematical model containing ordinary differential equations of the first order with initial data, algebraic equations, inequalities (linear and nonlinear), and logical expressions. Differential equations must be solved for the derivative. Parameters of differential

and algebraic equations as well as initial data may be specified precisely or as intervals.

Presently, UniCalc uses the conventional numerical approach to solving systems of ODEs. In this approach, the derivatives are approximated by finite differences, following which the procedure is iterated over the integration domain with a certain step. To approximate the derivatives, we use explicit and implicit Euler schemes, as well as explicit Runge-Kutta schemes of various orders. Thus, the system of algebraic-differential equations is reduced to a system of algebraic relations (possibly with interval parameters), and the algebraic system thus obtained is solved with the help of the basic method of the solver. The values obtained at some step are used as the initial values for the same system at the following step. Note that in the case of implicit methods we do not need the initial approximation.

The user can choose the finite difference scheme from the options offered by the system, the accuracy for automatic step selection or a constant integration step, and the result output points. The results are produced as arrays of intervals for each unknown function of the system. We want to point out that in the current version of the ODE-preprocessor, intervals for each function contain solutions of the approximated differential equations but not the original system of differential equations. The calculation process can be accompanied by a plot in which each approximated function is represented as a band-curve bounded by the upper and lower boundaries of intervals.

7 Numerical experiments

To estimate the efficiency of the solver and determine the range of possible applications, many problems have been tested, including linear and nonlinear systems of equations and inequalities, mixed systems, various integer problems, optimization problems, interval problems, systems of differential equations, etc.

Nonlinear systems. Among the numerous nonlinear systems solved with UniCalc (including almost all tests considered in papers [13, 14]), we want to point out the problem offered in [15] as a test problem. This problem

concerns combustion of propane in air to form ten products. The system includes eleven equations in eleven unknowns and is as follows:

$$
\begin{aligned}
f_1 &= n_1 + n_4 - 3 = 0 \\
f_2 &= 2n_1 + n_2 + n_4 + n_7 + n_8 + n_9 + 2n_{10} - R = 0 \\
f_3 &= 2n_2 + 2n_5 + n_6 + n_7 - 8 = 0 \\
f_4 &= 2n_3 + n_9 - 4R = 0 \\
f_5 &= K_5 n_2 n_4 - n_1 n_5 = 0 \\
f_6 &= K_6 n_2^{1/2} n_4^{1/2} - n_1^{1/2} n_6 \left(\frac{p}{n_T}\right)^{\frac{1}{2}} = 0 \\
f_7 &= K_7 n_1^{1/2} n_2^{1/2} - n_4^{1/2} n_7 \left(\frac{p}{n_T}\right)^{\frac{1}{2}} = 0 \\
f_8 &= K_8 n_1 - n_4 n_8 \left(\frac{p}{n_T}\right) = 0 \\
f_9 &= K_9 n_1 n_3^{1/2} - n_4 n_9 \left(\frac{p}{n_T}\right)^{\frac{1}{2}} = 0 \\
f_{10} &= K_{10} n_1^2 - n_4^2 n_{10} \left(\frac{p}{n_T}\right) = 0 \\
f_{11} &= n_T = \sum_{i=1}^{10} n_i
\end{aligned}
$$

where K_i, p, and R are constants. We need to find the physical solution of this system, i.e. such that all the n_i are positive.

It is noted in the paper that in the original formulation this is a hard problem and it was reduced to another equivalent system which was solved. In contrast to this, UniCalc easily solved the original problem.

Determining the initial intervals of the variables took 2.5 secs, and finding the final solution, about 3 mins on an IBM PC AT–286.

Optimization problems. Currently, UniCalc does not have any special facility to solve optimization problems. However, its algorithm is general enough to solve some problems of this type. Both problems of unconstrained optimization and optimization under restrictions are considered. The first 15 problems offered in [16] as test problems were successfully solved. The first test was to find a minimum of the Rosenbrock's function of order fifteen.

One way of stating an optimization problem for UniCalc is its mathematical statement. For example, to find a maximum of a polynomial in the UniCalc language we write

```
      (* Find a maximum of the function *)
f(x) := x^6 - 26 * x^3 + 45 * x^2 - 10 * x + 1;
fmax  = f(x) ;
      (* The first derivative is zero *)
dif(f(), x) = 0;
      (* At the point of maximum,          *)
      (* the second derivative is negative *)
dif(f(), x:2) < 0;
```

The results are $x = 1.2176$, $fmax = 11.86$.

Integer problems. UniCalc can be used to solve various integer problems which are difficult for other solvers, for example, integer optimization, Diophantine equations, integer factoring, etc.

Problems with subdefinite data. UniCalc is good to solve research problems and problems with imprecise data. For example, suppose we have the following problem.

A ball falling freely from height $h = 2$ meters collides elastically with a stationary plane inclined at angle α to the horizon. It is known that the inclination angle α is in the interval from 25 to 35 degrees, and the restitution coefficient Rc is in the range from 0.75 to 0.85. Determine the possible directions of the ball's velocity vector at the end of the collision such that the maximal height h_1 after the collision is greater then or equal to 0.5 meter. The system of equations is given below.

```
V  =  sqrt(2 * @g * h);
Vt =  V * sin(alpha);
Un = -Rc * V * cos(alpha);
Ut =  Vt;
ctg(beta) = abs(Un) / abs(Vt);
(Ut^2  + Un^2 ) * cos(alpha + beta)^2 = 2 * @g * h1;
```

```
h1  >=  0.5;
    (* Initialization *)
alphadeg := [25, 35];
Rc       := [0.75, 0.85];
h        := 2;
betadeg  := [0, 90];
    (* radian to degree conversion *)
convert(angle) := angle * 180 / @pi;
alphadeg        = convert(alpha);
betadeg         = convert(beta);
```

The solutions are:

```
alphadeg := [25, 27.39],   betadeg :=  [28.75, 32.14],
h1 = [0.5, 0.56],          Rc := [0.78, 0.85],
Un = [-4.82, -4.44],       Ut = [2.65, 2.88],
V  = [6.26, 6.26],         Vt = [2.65, 2.88].
```

Note that the values obtained for `Rc` and `alphadeg` differ from the initial ones due to the fact that UniCalc does not distinguish between input and output data. The initial values were updated according to the problem statement.

8 Conclusions and future research

This paper describes the implementation of the method of subdefinite calculations in the UniCalc solver. The practical applicability and usefulness of this mathematical apparatus have been proved by successfully solving both test problems and real problems. We consider that the solver described can be useful to a wide circle of users: students, engineers, designers, economists and research workers.

The UniCalc solver runs under MS-DOS/PC-DOS operating systems, version 3.0 and above on the computers IBM PC XT/AT or fully compatible. It requires 512 Kb of RAM and 600 Kb of free disk space. Such a configuration is suitable for up to 300 relations and variables. UniCalc does not require a math-coprocessor, but will use one if it is available to speed up the calculation.

Research on the apparatus of subdefinite models and UniCalc continues. Our next goals include:

1. Improving the solver's capabilities (for instance, introducing an object-oriented representation of models, improving computational efficiency, porting UniCalc to Unix and MS Windows).

2. Theoretic trends. These works include in particular:

   - Other representations of subdefinite data (for example, by multiintervals) with an appropriate modification of the calculation procedures;
   - Studying methods for implementing the considered algorithm on parallel computers and transputers;
   - Extending the notion of subdefiniteness to other objects (for example, subdefinite relations and subdefinite functions).

References

[1] Moore, R. E. *Interval analysis.* Englewood Cliffs, New Jersey, Prentice-Hall, 1966.

[2] Alefeld, G. and Herzberger, J. *Introduction to interval computations.* Academic Press, New York, 1983.

[3] Zahde, L. A. *Fuzzy sets.* Information and Control **8** (3) (1965), pp. 338–353.

[4] Leler, W. *Constraint programming languages. Their specification and generation.* Addison-Wesley, Reading, Massachussets, 1988.

[5] Hyvönen, E. *Constraint reasoning based on interval arithmetic.* In: "Proceedings of IJCAI–91", 1991, pp. 1193–1198.

[6] Narin'yani, A. S. *Subdefinite models and operations with subdefinite values.* Preprint 400, USSR Acad. of Sciences, Siberian Division, Computer Center, Novosibirsk (1982) (in Russian).

[7] Narin'yani, A. S. *Subdefiniteness in knowledge representation and processing systems.* Transactions of USSR Acad. of Sciences, Technical Cybernetics **5** (1986), pp. 3–28 (in Russian).

[8] Narin'yani, A. S. *Active data types for representing and processing of subdefinite information.* In: "Actual Problems of the Computer Architecture Development and Computer and Computer System Software", Novosibirsk, 1983, pp. 128–141 (in Russian).

[9] Telerman, V. V. *Active data types.* Preprint 792, USSR Acad. of Sciences, Siberian Division, Computer Center, Novosibirsk (1988) (in Russian).

[10] Narin'yani, A. S., Telerman V. V., and Dmitriev, V. E. *Virtual dataflow machine as vehicle of inference/computations in knowledge bases.* In: Jorrand, Ph., Sgurev, V. (eds.) "Artificial Intelligence II. Methodology, Systems, Application", North-Holland, 1987, pp. 149–155.

[11] Dmitriev, V. E. *Technological complex for producing problem-oriented S-processors.* In: "Designing Software Tools for Intelligent Problems",

USSR Acad. of Sciences, Siberian Division, Computer Center, Novosibirsk (1988), pp. 103–111 (in Russian).

[12] Kashevarova, T. P. and Semenov, A. L. *Solving subdefinite problems for systems of ordinary differential equations of the first order in the UniCalc system.* In: "All-Union Scientific-technical Conference 'Intelligent Systems in Machine Building'", abstracts, Samara, 1991, pp. 21–24 (in Russian).

[13] Kearfott, R. B. *Some tests of generalized bisection.* ACM Transactions on Mathematical Software **3** (1987), pp. 197–220.

[14] Vrahatis, M. N. *Solving systems of nonlinear equations using the nonzero value of the topological degree.* ACM Transactions on Mathematical Software **4** (1988), pp. 312–329.

[15] Meintjes, K. and Morgan, A. P. *Chemical equilibrium systems as numerical test problems.* ACM Transactions on Mathematical Software **2** (1990), pp. 143–151.

[16] More, J. J., Garbow, B. S., and Hillstrom, K. E. *Testing unconstrained optimization software.* ACM Transactions on Mathematical Software **1** (1981), pp. 17–41.

Novosibirsk Division of the Russian
Research Institute of Artificial Intelligence
pr. Lavrent'eva 6, Novosibirsk,
Russia, 630090
E-mail: `semenov@isi.itfs.nsk.su`