Working Paper

Mathematical Modelling of Dynamical Processes Under Interval Experimental Data

Svetoslav Markov

WP-91-004
April 1991

IIASA International Institute for Applied Systems Analysis □ A-2361 Laxenburg □ Austria
Telephone: +43 2236 715210 □ Telex: 079 137 iiasa a □ Telefax: +43 2236 71313

Mathematical Modelling of Dynamical Processes Under Interval Experimental Data

Svetoslav Markov

WP-91-004
April 1991

International Institute for Applied Systems Analysis □ A-2361 Laxenburg □ Austria
Telephone: +43 2236 715210 □ Telex: 079 137 iiasa a □ Telefax: +43 2236 71313

Foreword

In recent years, significant progress has been made in the guaranteed treatment of mathematical models. It concerns all phases between the process of modelling and computer processing. The paper presents the basic components of a methodology for computation with automatic result verification, involving interval data. A full control over the computational errors and the uncertainty in the data is achieved by using well-defined interval computer arithmetics and dynamic accuracy of the data representation. The approach and its impact on the development of numerical algorithms is illustrated by interval versions of the problems of polynomial interpolation and least-square approximation.

Alexander B. Kurzhanski
Chairman
System and Decision Sciences Program

Mathematical Modelling of Dynamical Processes Under Interval Experimental Data

Svetoslav Markov

A brief discussion on a new methodology for solving mathematical problems involving interval input data and for scientific computing with result verification is presented. Some examples for the impact of this methodology on particular mathematical modelling situations are given. A brief report on a newly developed program system MODYNA, which is designed according to the new methodology, is presented.

1 Introduction

The development of a new methodology for numerical treatment of mathematical problems involving interval-valued (and more general set-valued) input data and for scientific computing with result verification has a strong impact over the mathematical modelling of dynamic processes. Recently developed problem solving environments designed according to this new methodology enables us to solve many mathematical problems involving interval-valued input data, obtaining thereby highly accurate and guaranteed bounds for the true solution sets. This leads to the possibility of rigorous evaluation of the effects of the imperfect modelling and to a mathematically clean motivation for the rejection of an incorrect mathematical hypothesis in a particular modelling situation.

In traditional numerical analysis, numerical algorithms are formulated in terms of familiar arithmetic operations between real numbers, as defined in the textbooks on real analysis. However, real arithmetic is unrecognizable to computers: they cannot execute real arithmetic operations in general (and who can?). The disability of computers to execute the real-arithmetic operations prescribed by the traditional numerical algorithms has led to various and rather arbitrary realizations of the arithmetic and conversion procedures on various types of computers. As a consequence it often happens that an algorithm produces (sometimes completely) different results when run on different types of computers even when these computers operate with the same precision. This ridiculous situation contradicts the basic idea incorporated in the concept of algorithm, namely the strict and accurate definition of the whole computational processes.

Because of the above mentioned uncertainty and arbitrariness of the computational process, the users of numerical algorithms are faced with the tedious problem of establishing a reliable connection between the correct solutions of the problem and the computational results practically obtained when running the particular numerical algorithm on a computer. The estimation of the global computational error is usually done through laborious independent estimations of both the truncation and the round-off errors.

Some intuitive techniques that are often used in practice for the estimation of the rounding errors are:

i) computation of residuals (which are expected to be close to zero);

ii) repeating the computational process with slightly changed data and comparing the results with the previous one;

iii) repeating the computations in several various precisions (single, double, extended etc.) and comparing the results.

However, it can be shown [21] that none of techniques i) - iii) are reliable. Amongst other techniques of error control we should mention forward and backward analysis [41]. These techniques require the computation of a large number of error estimates and so-called condition numbers. We share the opinion (see [21], p.15) that they are rather complicated and are still of limited practical usage.

For many years it has been believed that some wonderful programming tools can be designed in such a way that the application of these tools on traditional numerical methods should produce safe and accurate results. In the last decade it has become clear that such software tools hardly exist. The investigations took a new path: not only new types of software-hardware tools are needed now, but also numerical methods of a completely new type have to be designed, and then, a revolutionary new methodology arose putting together the new numerical methods and the new software-hardware tools.

That is why we are now talking not just of new programming tools or of new numerical algorithms, but rather of a new methodology for scientific computing that has as its goal the design of problem solving environments with automatic result verification, providing thus full control over the effects of the computational errors and the uncertainties in the data. The basic features of this methodology can be found in the recent volumes of the journal *Computing* (papers [4], [7], [22], [23], [32], and [36] are from such volumes).

The new methodology makes use of, e.g.

- a mathematically precise definition of computer arithmetic operations in all computational spaces (real and complex numbers, intervals, vectors, matrices, interval vectors, functions etc.) and the implementation of these operations in the computer,
- suitable formulation of the numerical algorithms using the above mentioned extended arithmetic facilities. The usage of a well-defined computer arithmetic in all necessary computational spaces allows the construction of numerical algorithms that always produce well-defined intermediate and final results depending only on the chosen computational precision.
- new algorithmic techniques needed to comprise the computational errors and to compute with interval input data (or a more general type of set-valued data).

The new type of numerical algorithms possess (some of) the following properties:

- problems involving interval input data (or other types of set-valued data) can be successfully handled;
- the numerical algorithms produce guaranteed bounds for the ideal solution of the numerical problem;
- these bounds are the smallest possible within the chosen precision of the floating-point screen,
- the above mentioned bounds can be made arbitrarily sharp (that is close to the true ones) by using the so-called staggered correction format (STC format) techniques [36] in order to serve as input data for exceptionally ill-conditioned mathematical tasks.

In the frames of the new methodology some new mathematical tools such as interval analysis and computer arithmetic [2], [19], [29], [34] and some new application of well known iteration techniques, differential inequalities, monotonicity theorems, fixed-point theorems etc. [1-12, 32, 35, 36] have been developed. The new standards for the implementation of computer and interval arithmetic on arithmetic processors provide the necessary support for the computational approach (see [8], [15], [16], [42] and the IMACS-GAMM resolution of computer arithmetic ([39], p. 301)). The design and the practical use of the new problem solving environments have a strong impact on mathematical modelling since they permit an evaluation of the effects of imperfect modelling and a reduction of the computational errors below certain (well-known in advance) bounds.

In numerical computations, two goals have always been pursued: guaranteed accuracy and speed. Here we will not be concerned with problems related to acceleration of the computations. Let us mention that the new methodology for safe and accurate computations can be successfully combined with parallelization techniques [38].

2 Numerical algorithms with result verification for mathematical problems involving interval input data.

Let us make some general considerations in relation to the numerical solution of a scientific problem. A solution y of a well defined mathematical problem can be considered as a function (or operator) of the input data x for some domain D of variation of the data: $y = f(x)$. For simplicity let us consider the situation when f is a function (for instance, y can be a real number, D can be a subset of R^n etc.). (The situation when x and y are functions and f is an operator is studied in [17].) When using a conventional numerical procedure for the computation of y we traditionally speak about three possible types of errors:

i) errors in the input data x;

ii) errors from the numerical method (e.g. due to truncation of an infinite iterative process);

iii) errors from the finite representation of floating-point numbers.

The origin of the errors from the first type may lie in the imprecise experimental measurements. Usually the experimental scientist is able to read-off an interval for the true value of the measured quantity. In some cases this interval may contain the true value with a guarantee. It may turn out that this is a very common situation, despite of the fact that the usual praxis is to read-off numerical data. The new developments should encourage the experimenters to read-off interval-valued experimental data. We shall further assume that for the input data we are given some (interval) bounds. In such a situation there is not much reason to talk about "errors" in the input data (especially when the bounds are guaranteed) so we shall instead speak of interval input data (or more general set-valued data like ellipsoids, etc.)

Even if the bounds for the input data are small, it may happen that they cause large deviations in the final results (by ill-conditioned problems). That is why the safest way to treat such problems is to consider them as set-valued problems never mind how small the input intervals are. A problem $P(X)$ involving interval input data X is thus considered as a set of problems $P(x)$ with numerical data x, such that $x \in X$, i.e. $P(X) = \{P(x) : x \in X\}$. The solution $Y(X)$ of $P(X)$ is then by definition the set of solutions $Y(x)$ of $P(x)$ whenever $x \in X$, i.e.

$Y(X) = \{Y(x) : x \in X\}$. An efficient tool for the treatment of problems involving interval input data is interval analysis [2], [23], [25], [28-30].

In the same manner we can treat the errors due to the necessarily finite representation of numerical input data (e.g. 1/3 is represented in a base 10 floating-point system by an interval of the form [0.33 . . .33, 0.33 . . . 34]). In such cases we replace the numerical input data by interval data, but now we also may interfere to make these bounds as tight as we wish (using e.g. extended precision formats). This situation takes an intermediate place between the first and the other two error type situations. Here we have a problem with set-valued input data, but the bounds for these data can be made arbitrary small, e.g. by using the STC-formating technique.

Let us turn our attention to the errors of types ii) and iii). To this end we may assume that the argument x takes values from the computational space S so that there are no errors from the first type. Our ultimate goal is to design a computational algorithm that do not introduce any unpredictable errors of types ii) and iii), that is to obtain a solution that lies within known bounds. For instance the numerical algorithm may produce the optimal (best) approximation in the floating point system S of the real result y as defined by the equality $y = f(x)$ in real arithmetic. When talking at this point about the value of y, we do not mean the result of some computational process producting y; we just define the real number y by the equality $y = f(x)$ in the sense of analysis. Since we do not have real numbers in the computer at our disposal, the best that could be expected is to obtain the shortest machine interval on the floating-point screen that contains the ideal result $y = f(x)$. In this case we may say that y is computed within the maximum accuracy allowed by the system S.

The new methodology for scientific computing and for design of numerical methods with result verification can not be outlined sufficiently well without some elementary knowledge of computer and interval arithmetic. These two mathematical tools are a substantial part of the necessary mathematical background for the construction of such methods.

3 Computer and interval arithmetic

In what follows $S = S(b,p)$ denotes the standard set of floating-point numbers expressible in a base b number system with a mantissa of some fixed length p (called precision), that is $S(b,p)$ is the set of all real numbers of the form $m \times b^n$, where n is an arbitrary integer but m is an integer of absolute value less than b^p [19].

Let us denote the shortest machine interval containing the real result y by $[\bigtriangledown y, \bigtriangleup y] =: \diamond y$, wherein $\bigtriangledown y = \max\{x : x \in S, x \leq y\}, \bigtriangleup y = \min\{x : x \in S, x \geq y\}$.

Note that the interval $\diamond y$ does not contain any computer numbers as inner points. We shall further consider the symbols ∇, Δ as functions (called also rounding functions), mapping R into itself. The screen S is the range of these functions. The rounding functions ∇, Δ are monotone projections. A function $O : R \longrightarrow S, S \subset R$ is called a monotone projection, if [19]:

i) $0(0(x)) = 0(x)$ for all $x \in R$;

ii) $x \leq y$ implies $O(x) \leq O(y)$ for all $x, y \in R$;

iii) $O(0) = 0; O(1) = 1$.

We see that the result of a computer arithmetic operation or procedure should be the smallest floating-point interval containing the true result. Since we further have to operate with such interval results we thus arrive to the necessity of using interval arithmetic operations. After introducing interval arithmetic, we shall be able to consider the rounding $\diamond$ as a rounding interval-valued function mapping IR (and, in particular, R) onto IS, where IR, resp. IS, is the set of all intervals with end-points in R resp. in S, symbolically, $\diamond : IR \longrightarrow IS$. In this case the order relation should be taken to be that of inclusion.

The rounding function can be defined in an abstract way as a mapping satisfying the above three relations over an ordered field or over more general algebraic structures. The properties of O in such setting are studied in detail in [19].

From a practical point of view, it is important that a monotone projection O always produces machine numbers or intervals of best (optimal) approximation. The rounding O possesses the above mentioned optimality property. By means of the rounding O we can define the basic computer operations for addition, multiplication, scalar product and others in the corresponding computational real arithmetic results $x + y, xy, u \circ v$. This principle of definition was adopted for the arithmetic operations in S in the IEEE resolutions on computer arithmetic from 1979 and 1985 [8] and was further extended for higher computational spaces (such as S^n) in the IMACS-GAMM resolution on computer arithmetic from 1987 (see e.g. [39], p.301).

For simplicity we shall assume the computational spaces to be the corresponding floating-point spaces (and not the finite spaces of machine elements). In programmer's slang we may say that for the moment we shall not be interested in overflow and underflow. In practice overflow and underflow occur rarely and does not cause so much trouble.

Computer arithmetic operations. Let $x, y \in S$ be floating-point numbers and $u = (u_1, u_2, \cdots, u_n), v = (v_1, v_2, \cdots, v_n) \in S^n$ are floating-point vectors. We define the computer operations corresponding to the real operations $x + y, xy, u \circ v$ for addition, multiplication and scalar product to be the shortest machine intervals containing the real (ideal) result of the corresponding operation:

i) *addition* in S: $x \diamondsuit y = \diamond(x+y) = [\triangledown(x+y), \triangle(x+y)] = [x \triangledown y, x \triangle y]$;

ii) *multiplication* in S : $x \otimes y = \diamond(xy) = [\triangledown(xy)] = [\triangledown(xy), \triangle(xy)] = [x \triangledown y, x \triangle y]$;

iii) *scalar product* in S^n : $u \bullet v = \diamond(u \circ v) = [\triangledown(u \circ v), \triangledown(u \circ v)]$, wherein $u \circ v = \sum_{i=1}^{n} u_i v_i$.

We note that all the above intervals contain no elements of the floating-point system S inside (as inner points) and in this sense these results are optimal (i.e. with maximum accuracy).

Example 1. Compute $(x \diamondsuit y) \diamondsuit z$ and $x \diamondsuit (y \diamondsuit z)$ for $x = 5$, $y = 10^7$, $z = -10^7$ in the computational space $S(10, 7)$.

Solution. We have

$$\triangledown(x+y) = \triangledown 10000005 = 10000000, \quad \triangle(x+y) = \triangle 10000005 = 10000010,$$

and therefore $x \diamondsuit y = [10000000, 10000010]$.

We now compute $(x \diamondsuit y) \diamondsuit z$ which is

$$(x \diamondsuit y) \diamondsuit z = [(x \triangledown y) \triangledown z, (x \triangle y) \triangle z].$$

by the definition of interval addition (see the definition below). From $(x \triangledown y) \triangledown z = 10^7 \triangledown -10^7 = 0$, $(x \triangle y) \triangle z = 10000010 \triangle -10000000 = 10$ we obtain $(x \diamondsuit y) \diamondsuit z = [0, 10]$. Similarly, we compute $x \diamondsuit (y \diamondsuit z) = 5$.

This example shows that it is useful to define operations between intervals. This will be outlined in the next section. The example also shows that addition in S does not fulfill associative low.

Example 2. Compute in $S^3(10, 7)$ the scalar product $u \bullet v$ with $u = (5, 10^3, 10^4), v = (1, 10^4, -10^3)$.

Solution. The ideal result is $u \circ v = 5 \times 1 + 10^3 \times 10^4 + 10^4 \times (-10^3) = 5 + 10^7 - 10^7 = 5$, and therefore $u \bullet v = 5$.

Remark. The absence of the operation $\bullet$ in $S^3(10, 7)$ could lead to a very bad approximation of the correct value of the scalar product $u \circ v$. Indeed, without this operation for the scalar product we have $(5 \diamondsuit 10^7) \diamondsuit (-10^7) = [0, 10]$ (see Example 1). Note also that the scalar product in S^n provides the space S with an exact sum of n summands, since $\sum_{i=1}^{n} u_i = u \circ v$, with $u = (u_1, u_2, \cdots, u_n), v = (1, 1, \cdots, 1)$.

We defined two operations in the set S (addition and multiplication), and an operation in S^n (scalar product). Similarly two operations called subtraction and division are defined in S that are inverse operations to addition and multiplication.

For the structural properties of computer arithmetic the reader can consult [19]. For the software and hardware implementation of the computer arithmetic operations, including the scalar product, see [8, 19, 21]. The application of the computer arithmetic operations for the design of safe and accurate numerical algorithms for scientific computation is considered in [21, 26, 28, 29, 32, 33].

Interval arithmetic. Let us now briefly consider the interval arithmetic. The simplest interval arithmetic operations are the arithmetic operations in the set IR of the intervals on the real line. If $A, B \in IR$ with $A = [\underline{a}, \bar{a}]\,, B = [\underline{b}, \bar{b}]$, we define

$$A + B = [\underline{a} + \underline{b}, \bar{a} + \bar{b}]\,,$$

$$A \oplus B = \begin{cases} [\underline{a} + \bar{b},\ \bar{a} + \underline{b}]\,, \text{ if } w(A) \geq w(B)\,, \\ [\bar{a} + \underline{b},\ \underline{a} + \bar{b}]\,, \text{ if } w(A) \leq w(B)\,. \end{cases}$$

wherein w denotes the width of the corresponding interval, that is $w(A) = \bar{a} - \underline{a}$.

We also define the scalar multiplication by

$$\alpha \bullet A = \alpha A = \begin{cases} [\alpha\underline{a}, \alpha\bar{a}]\,, & \text{if } \alpha \geq 0, \\ [\alpha\bar{a}, \alpha\underline{a}]\,, & \text{if } \alpha \leq 0. \end{cases}$$

For $\alpha = -1$ the above formula gives $-1 \bullet A = [-\bar{a}, -\underline{a}]$. This interval will be denoted further by $-A$. We then denote $A - B = A + (-B)$, $A \ominus B = A \oplus (-B)$. The operation $\ominus$ plays an important role in the differential calculus for interval functions. Because of this reason we have exchanged the notations "$-$" and "$\ominus$" in our publications on this topic [23-26].

We see that the two operations for addition generate two operations for subtraction. This is natural by the observation that the interval space has a richer structure that the space of real numbers. Let us note that the interval produced by the operation "$\ominus$" is narrower than the interval result produced by the operation "$-$"; also the operation "$\oplus$" delivers a sharper result than the operation "$+$".

Let the end-points of an interval A be α and β, that is

$$A = \begin{cases} [\alpha, \beta]\,, & \text{if } \alpha \leq \beta, \\ [\beta, \alpha]\,, & \text{otherwise.} \end{cases}$$

The above formula can also be written as $A = [\alpha \vee \beta]$ (read: "the end-points of the interval A are α and β "). For the definition of multiplication it is useful to have a special notation for the end-point of an interval A that is closer to zero than the other end-point. Let us denote the closer to zero end-point by a^{+0} and the other one by a^{-0}. We thus have $A = [a^{+0} \vee a^{-0}]\,, |a^{+0}| \leq |a^{-0}|$.

We shall introduce next two operations for multiplication of intervals that do not contain zero (an extension of the definition for arbitrary intervals can be found in [9], [23]).

Denote by IR^* the set of all intervals A such that $0 \notin A$. For $A, B \in IR^*$ we define $A \times B = [a^{+0}b^{+0} \vee a^{-0}b^{-0}]$, $A \otimes B = [a^{+0}b^{-0} \vee a^{-0}b^{+0}]$,

If $B = [b^{+0} \vee b^{-0}] \in IR^*$, denote $B^{-1} = 1/B = [(1/b^{+0}) \vee (1/b^{-0})]$. The two operations for multiplication generate the following two operations for division: $A/B = A \times B^{-1}$, $A \oslash B = A \otimes B^{-1}$.

We note that the operations $\otimes$ and $\oslash$ produce narrower interval results than the corresponding operations $\times, /$.

Let us pay some attention to the fact that the operations $-, \ominus, /, \oslash$ are composite operations defined by

$$(*)\ A - B = A + (-B), A \ominus B = A \oplus (-B), A/B = A \times B^{-1}, A \oslash B = A \otimes B^{-1}\ ,$$

and should not be taken in considerations as basic operations (e.g. one should write $< IR, +, \times >$ and not $< IR, +, -, \times, / >$ as it is often done in the literature). Because of (*) we shall further call the operations $-, \ominus, /, \oslash$ quasiinverse to the corresponding operations $+, \oplus, \times, \otimes$.

Remark. We can not assert that the operations $-, \ominus, /, \oslash$ are inverse to the operations $+, \oplus, \times, \otimes$ in the usual sense. For instance, the operation $-$ is not inverse to the operation $+$, since $X = A - B$ is not a solution of $A + X = B$ in general. The same holds true for the rest of the operations: $\ominus, /, \oslash$.

The interval arithmetic structure $< IR, +, \oplus, \times, \otimes >$, which also includes the quasiinverse operations $-, \ominus, /, \oslash$, is known as *extended interval arithmetic* . It is a rich algebraic structure which is very useful for the interval arithmetic presentation of interval extensions of real functions, and, as a consequence, for the treatment of numerical problems involving interval input data [9], [23], [24], [25].

The elimination of the operations $\oplus, \otimes$ from the extended interval arithmetic leads to a simpler interval arithmetic structure $< IR, +, \times >$, which also includes the quasiinverse operations $-, /$. This structure is known as *standard interval arithmetic* [2], [28]. It is very effective for the construction of interval arithmetic inclusions of interval extensions, and is thus very useful for the automatic validation processes.

A short survey of various interval arithmetic structures can be found in [34]. In what follows we give a brief comparison between the interval arithmetic structures $< IR, +, \bullet >$ and $< IR, +, \oplus, \bullet >$ as substructures of the standard and extended interval arithmetic, respectively.

Below we give a list of basic properties of $< IR, +, \oplus, \bullet >$.

For $A, B, C \in IR$, $\alpha, \beta \in R$ we have:

1) $A + B = B + A$;

1a) $A \oplus B = B \oplus A$;

2) $\alpha(B + C) = \alpha B + \alpha C$;

2a) $\alpha(B \oplus C) = \alpha B \oplus \alpha C$;

3) $\alpha\beta \geq 0$ implies $(\alpha + \beta)C = \alpha C + \beta C$;

3a) $\alpha\beta \leq 0$ implies $(\alpha + \beta)C = \alpha C \oplus \beta C$;

4) $\alpha(\beta C) = (\alpha\beta)C$;

5) $1 \bullet A = A$;

6) $0 \bullet A = 0$;

7) $(A + B) + C = A + (B + C)$;

7a)

$$(A + B) \oplus C = \begin{cases} A + (B \oplus C), \text{if } w(B) \geq w(C), \\ A \oplus (B \oplus C), \text{if } w(B) \leq w(C); \end{cases}$$

7b)

$$(A \oplus B) + C = \begin{cases} A \oplus (B + C), \text{if } w(B) \geq w(A), \\ A + (B \oplus C), \text{if } w(B) < w(A), w(B) < w(C), \\ A \oplus (B \oplus C), \text{if } w(B) < w(A), w(B) \geq w(C); \end{cases}$$

7c)

$$(A \oplus B) \oplus C = \begin{cases} A \oplus (B \oplus C), \text{if } w(B) \geq w(A), w(B) \geq w(C), \\ A + (B \oplus C), \text{if } w(B) \geq w(A), w(B) < w(C) \\ A \oplus (B + C), \text{if } w(B) < w(A). \end{cases}$$

Now, to obtain a list of properties of the interval space $< IR, +, \bullet >$ we should exclude the assertions involving the operation $\oplus$: these are equalities 1a), 2a) , 3a), 7a), 7b) and 7c).

After defining arithmetic operations on IR , we can extend these operations for other interval objects: interval vectors, interval matrices, interval functions etc. A possible approach in this direction is outlined in [24]. On the basis of interval arithmetic an extensive mathematical theory for the study of interval objects called *interval analysis* has been developed. Interval analysis is a rapidly developing theory and there is already vast literature on this subject. From a practical point of view, the main application of interval arithmetic and interval analysis is in the treatment of numerical problems involving interval input data and in the organization of

iterative procedures involving automatic validation. According to [21], "interval arithmetic is the only computational tool so far available that incorporates guarantees as part of the basic computational process."

By means of computer and interval arithmetic, one can define arithmetic operations in the set of all intervals on the floating-point screen S [19]. For instance, addition of $A = [\underline{a}, \bar{a}], B = [\underline{b}, \bar{b}] \in IS$ is defined by means of $A + B = [\underline{a} \bigtriangledown \underline{b}, \bar{a} \bigtriangleup \bar{b}]$.

4 Dynamic accuracy and STC format

Let us first discuss an example. Consider the matrix:

$$A = \begin{pmatrix} 1+\delta & 1 \\ 1 & 1 \end{pmatrix},$$

where $\delta = 1/30$. Assume that we have to represent the matrix in the floating-point system S in order to solve a linear problem of the form $Ax = b$. For simplicity assume that $S = S(10, 2)$.

Then the sharpest interval matrix $\diamond A$ with elements of IS , which contains the matrix A is

$$\diamond A = \begin{pmatrix} [1.0, 1.1] & 1 \\ 1 & 1 \end{pmatrix}.$$

Since $\det(A) = \delta > 0$ and therefore A is not singular for an arbitrary right hand side b, the linear system $Ax = b$ has a unique solution. However, the interval matrix $\diamond A$ contains the singular matrix A^*

$$A^* = \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}.$$

Thus the set of problems, corresponding to the interval matrix $\diamond A$ (this set of problems is usually denoted simply by $\diamond Ax = b$) contains a problem that has no solution (the problem $A^*x = b$, corresponding to the singular matrix A^*). In such a situation we may say that the problem represented in the computer has no solution (although the original problem has a well defined solution. Stetter [35, 4] proposed a following way to circumvent this obstacle. Let a be a numerical input data of the problem we are solving (think of $a = 1 + \delta$ in the above example). If the sharpest interval $\diamond a$ with endpoints in S such that $a \in \diamond a$ is too large and makes our problem unsolvable, we then take a sufficiently shorter interval, presented in the form

$(**)$ $a_1 + a_2 + \cdots + a_{r-1} + [\underline{a}_r, \bar{a}_r]$,

where the numbers $a_1, a_2, \cdots a_{r-1}, \underline{a}_r$ and $\bar{a}_r$ belong to the computational system S.

The interval (**) is usually coded as $(a_1, a_2, \cdots, a_{r-1}, [a_r])$, where $[a_r]$ is an abreviation for $[\underline{a}_r, \bar{a}_r]$. We now see that although all a_i 's in the form (**) are within S , the endpoints of the interval (**) are not from S. The representation (**) is called staggered correction representation (STC-representation, STC-format). This representation does not need special arithmetic utilities and gives us the possibility to extend the precision as much as we like. For the matrix A from the above example the STC representation with $r = 2$, has the form $A^1 + [A^2]$, namely

$$C = A^1 + \left[A^2\right] = \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix} + \begin{pmatrix} [.033, .034] & 0 \\ 0 & 0 \end{pmatrix}.$$

The above interval matrix C is much shorter than the interval matrix $\diamond A$; the important fact now is that the problem $Cx = b$ has a well defined solution.

5 Interval mathematical problems. Interval interpolation and approximation

In this section we consider two simple examples of interval mathematical problems: polynomial interpolation and least-square approximation in the presence of interval input data.

In the situation when we have guaranteed intervals from the observations of a stochastic variable it may be useful to apply the least-square approximation method directly to the interval input data, obtaining thus (as usually in the interval analysis) the set of all approximations to the numeric data varying in the given intervals.

Interval least-square approximation. We first recall some well-known results related to the least-square approximation method under numeric data, considering the most simple linear one-dimensional case.

The coefficients a and b of the line

(1) $$l : \eta = a\xi + b$$

that fits to the input data $(x, y), x = (x_1, x_2, \cdots, x_N) \in R^N$, $y = (y_1, y_2, \cdots, y_N) \in R^N$ so that $\sum (ax_i + b - y_i)^2$ is minimal, are determined by the system

$$\left(\sum x_i^2\right) a + \left(\sum x_i\right) b = \sum x_i y_i,$$

$$\left(\sum x_i\right) a + Nb = \sum y_i,$$

wherein $\sum$ means summation from 1 to N. Denoting $\bar{x} = (\sum x_i)/N$, $\bar{y} = (\sum y_i)/N$ and dividing the second equation into N we have

$$\left(\sum x_i^2\right) a + N\bar{x}b = \sum x_i y_i,$$

(2)

$$\bar{x}a + b = \bar{y}.$$

The determinant of (2) will be further denoted by

$$S_{xx} = \sum x_i^2 - N\bar{x}^2 = \sum (x_i - \bar{x})^2 > 0.$$

The slope a of the line l is

$$a = (1/S_{xx})\left(\sum x_i y_i - N\bar{x}\bar{y}\right)$$

which can also be written

$$a = \left(\sum x_i y_i - \bar{x}\sum y_i\right)/S_{xx} = \left(\sum (x_i - \bar{x})\, y_i\right)/S_{xx} \stackrel{Df}{=} S_{xy}/S_{xx}.$$

For b we compute $b = \bar{y} - a\bar{x} = \bar{y} - a\bar{x} = \bar{y} - (S_{xy}/S_{xx})\,\bar{x}$, so that (1) obtains the form

(3) $$l : \eta = \eta(x, y; \xi) = (S_{xy}/S_{xx})(\xi - \bar{x}) + \bar{y}$$

showing that l passes through the point $(\bar{x}, \bar{y})$.

In what follows we shall also need the following presentation of the regression line

$$\begin{aligned} l : \eta &= a(\xi - \bar{x}) + \bar{y} \\ &= (S_{xy}/S_{xx})(\xi - \bar{x}) + \bar{y} \\ &= (1/S_{xx})(\sum (x_i - \bar{x})\, y_i)(\xi - \bar{x}) + (\sum y_i)/N \\ &= \sum ((x_i - \bar{x})(\xi - \bar{x})/S_{xx} + 1/N)\, y_i, \end{aligned}$$

or

(4) $$l : \eta = \sum \gamma_i(\xi)\, y_i,$$

wherein the functions

(5) $$\gamma_i(\xi) = \gamma_i(x; \xi) = (x_i - \bar{x})(\xi - \bar{x})/S_{xx} + 1/N,\ i = 1, 2, \cdots, N,$$

depend only on x (not on y !) .

Since γ_i is linear, it may have at most one zero. If a point x_i coincides with $\bar{x}$, then $\gamma_i = 1/N > 0$. Consider the case $x_i \neq \bar{x}$. Then $\gamma_i(\xi)$ has a slope $(x_i - \bar{x})/S_{xx}$ such that

$$(x_i - \bar{x})/S_{xx} \begin{cases} > 0, \text{ if } x_i < \bar{x}, \\ < 0, \text{ if } x_i > \bar{x}. \end{cases}$$

Let $x_i < \bar{x}$ for $i = 1, 2, \cdots, j$ and $x_i > \bar{x}$ for $i = j+1, \cdots, N$. Denoting by ξ_i the zero of the linear function $\gamma_i(\xi)$, i.e.

$$\gamma_i(\xi_i) = (x_i - \bar{x})(\xi_i - \bar{x})/S_{xx} + (1/N) = 0,$$

we have

$$\xi_i = \bar{x} + S_{xx}/(N(\bar{x} - x_i)). \tag{6}$$

The ordering of the x_i 's with respect to $\bar{x}$ imply corresponding ordering of the ξ_i's defined by (6). Namely,

$$x_1 < x_2 < x_3 < \cdots < x_j < \bar{x} < x_{j+1} < \cdots < x_N$$

$$\Rightarrow \xi_{j+1} < \xi_{j+2} < \cdots < \xi_N < \bar{x} < \xi_1 < \xi_2 < \cdots < \xi_{j-1} < \xi j.$$

These relations remain true also for $x_j = \bar{x}$, providing that in this case ξ_i is understood as ∞ , so that we could write

$$x_1 < x_2 < x_3 < \cdots < x_j \leq \bar{x} < x_{j+1} < \cdots < x_N$$

(7)

$$\Rightarrow \xi_{j+1} < \xi_{j+2} < \cdots < \xi_N < \bar{x} < \xi_1 < \xi_2 < \cdots < \xi_{j-1} < \xi_j.$$

According to (7), the minimal interval I containing the ξ_i's is

$$I = \begin{cases} [\xi_{i+1}, \xi_j], \text{ if } x_j \neq \bar{x}, \\ [\xi_{j+1}, \xi_{i-1}], \text{ if } x_j = \bar{x}. \end{cases}$$

We shall adopt the notations $D_\circ, D_1, \cdots, D_N$ for the intervals with end-points ξ_i as follows:

$$D_\circ = [\xi_N, \xi_1], D_1 = [\xi_1, \xi_2], \cdots, D_{j-1} = [\xi_{j-1}, \xi_j], D_j = [\xi_j, \infty],$$

$$D_{j+1} = [-\infty, \xi_{j+1}], D_{j+2} = [\xi_{j+1}, \xi_{j+2}], \cdots, D_N = [\xi_{N-1}, \xi_N].$$

Let us now compute the sign of $\gamma_i(\xi)$ in the interval D_k. In $D_o = [\xi_N, \xi_i]$ all $\gamma_i(\xi), i = 1, 2, \cdots, N$ have positive signs. We shall call D_o the "central interval". In the remaining intervals we have

i) to the right of D_o, that is for $\xi \in D_k, 1 \leq k \leq j$:

$$(8a) \qquad \text{sign}\gamma_i(\xi) = \begin{cases} -, i = 1, \cdots, k \\ \\ +, i = k+1, \cdots, N \end{cases} = \text{sign}(i - k - 1/2), i = 1, \cdots, N.$$

ii) to the left of D_o , that is for $\xi \in D_k, j+1 \leq k \leq N$.

$$(8b) \qquad \text{sign}\gamma_i(\xi) = \begin{cases} +, i = 1, \cdots, k-1 \\ \\ -, i = k, \cdots, N \end{cases} = \text{sign}(k - i - 1/2), i = 1, \cdots, N.$$

In the most right interval D_j (or D_{j-1} if $x_j = \bar{x}$) we have

$$\text{sign}\gamma_i(\xi) = \begin{cases} -, i = 1, \cdots, j \\ \\ +, i = j+1, \cdots, N \end{cases} = \text{sign}(i - j - 1/2), i = 1, \cdots, N.$$

In the most left interval D_{j+1} we have inversely

$$\text{sign}\gamma_i(\xi) = \begin{cases} +, i = 1, \cdots, j \\ \\ -, i = j+1, \cdots, N \end{cases} = \text{sign}(j - i - 1/2), i = 1, \cdots, N.$$

Let us now discuss the least square approximation method in the situation when interval-valued experimental data are provided for the true values of the observation.

Assume now that we are given N numbers $(x_1, x_2, \cdots, x_N) = x \in R^N$ such that $x_1 < x_2 < \cdots < x_N$ and N intervals $(Y_1, Y_2, \cdots, Y_N) = Y \in IR^N$ (IR^N is the set of N -dimensional interval vectors). Let $y \in R^N$ be such that $y \in Y$, and $\eta(x, y)$ be the regression linear function (3) generated by the input data (x, y) . Denote by $\mathcal{L}$ the family of all regression linear functions $\eta = \eta(x, y)$, generated by the input data x, y , whenever the numeric vector $(y_1, y_2, \cdots, y_N) = y$ varies in the interval vector $(Y_1, Y_2, \cdots, Y_N) = Y$, that is the set

$$(9) \qquad \mathcal{L}(x, Y) = \{\eta(x, y) : y \varepsilon Y\}.$$

Denote by L the set-valued function, corresponding to $\mathcal{L}$, defined for $\xi \in R$ by $L(\xi) = \{\eta(x, y; \xi) : y \in Y\}$.

PROBLEM A. Compute the set-valued function L or an inclusion for this function.

SOLUTION. We shall first consider the easier problem of computing an inclusion for L .

According to (3) the line $l(x, y)$ generated by x, y is the line passing through the point $m = (x, y) = (1/N)\sum y_i$ and having as slope $a = a(x, y) = S_{xy}/S_{xx} = (\sum (x_i - \bar{x})\, y_i)$. As y_i vary in $Y_i, i = 1, 2, \cdots, N$, the point m varies in the segment $M = (\bar{x},\ \bar{Y} = (1/N)\sum Y_i)$ and the slope a varies in $A = A(x, Y) = (\sum (x_i - \bar{x}) Y_i) = S_{xY}/S_{xx}$. The sets A and M are obtained from the variation of the y_i's. If we consider them as independent, we may construct the interval linear function

$$\hat{L} = A(x, Y)(\xi - \bar{x}) + \bar{Y} = \{a(\xi - \bar{x}) + \bar{y} : a \in A, \bar{y} \in \bar{Y}\}$$

that contains L. Since the parameters a and $\bar{y}$ are strongly dependent we shall obtain only rough bounds for L.

We may easily obtain the exact interval hull of L in explicit form if we use the representation (4) for l, that is $\eta = \sum \gamma_i(\xi)\, y_i$. Indeed, we have for every fixed ξ

$$\begin{aligned} L(\xi) &= \{\textstyle\sum \gamma_i(\xi)\, y_i : y \in Y\} \\ &= \textstyle\sum \gamma_i(\xi)\, Y_i \\ &= \left[\textstyle\sum \gamma_i(\xi)\, y_i^{-\operatorname{sign}\gamma_i(\xi)},\ \sum \gamma_i(\xi)\, y_i^{+\operatorname{sign}\gamma_i(\xi)}\right] \\ &= [l^-(\xi), l^+(\xi)]. \end{aligned} \tag{10}$$

In the above formula the end-points of the intervals Y_i are denoted by $y_i^- \le y_i^+$ so that $Y_i = \left[y_i^-, y_i^+\right], i = 1, 2, \cdots, N$ and $\operatorname{sgn}\gamma_i(\xi)$ means "+", if $\gamma_i(\xi) \ge 0$, and "−" , if $\gamma_i(\xi) \le 0$; also y_i^{--} means y_i^+ (right end-point) and y_i^{-+} means y_i^- (left end-point). Formula (10) shows that $L(\xi)$ is an interval at every ξ, so that L may be considered as interval function. The interval function L gives the exact interval bounds of L

$$L(\xi) = L(x, Y; \xi) = \sum \gamma_i(\xi)\, Y_i,$$

where, according to (5)

$$\gamma_i(\xi) = (x_i - \bar{x})(\xi - \bar{x})/S_{xx} + 1/N,\ i = 1, 2, \cdots, N.$$

We see that formula (10) gives the boundaries $l^-(\xi)$, $l^+(\xi)$ of the interval function $L(\xi)$. By means of interval arithmetic the interval function $L(\xi)$ is expressed in the simple form $L(\xi) = \sum \gamma_i(\xi)\, Y_i$.

GEOMETRICAL MEANING. To see the geometrical meaning of the expression for $L(\xi)$ we have to know the signs of $\gamma_i(\xi)$. These signs are constant in the intervals D_k defined by

$$D_0 = [\xi_N, \xi_1] \ni 0,\ D_i = [\xi_i, \xi_{1+i}], i = 1, 2, \cdots, j-1;$$

$$D_j = [\xi_j, \infty),\ D_{j+1} = (-\infty, \xi_{j+1}), D_i = [\xi_{i-1}, \xi_i], i = j+2, \cdots, N,$$

but are different for the different intervals D_k according to formulas (8).

According to (8) the functions $\gamma_i(\xi), i = 1, \cdots, N$, have positives signs in the "central" interval $D_0 = [\xi_N, \xi_1]$. In the other intervals $D_1, D_2, \cdots, D_j, D_{j+1}, \cdots, D_N$ we have

$$\text{sign}\gamma_i(\xi) = \begin{cases} -,\ i = 1, 2, \cdots, k; \\ +,\ i = k+1, \cdots, N;\ \xi \in D_k,\ i \leq k \leq j, \end{cases}$$

and, respectively,

$$\text{sign}\gamma_i(\xi) = \begin{cases} +,\ i = 1, 2, \cdots, k-1; \\ -,\ i = k, \cdots, N;\ \xi \in D_k,\ j+1 \leq k \leq N. \end{cases}$$

We see that in every fixed interval D_k the boundaries l^-, l^+ of the set L are segments of the regression lines with respect to certain end-points of the input intervals $Y_1, Y_2, \cdots Y_N$. So, in the "central" interval D_0 the boundary regression line l^+ is generated by the set of all right end-points of $Y_1, Y_2, \cdots Y_N$ and the boundary line l^- is generated by the set of all left end-points of $Y_1, Y_2, \cdots Y_N$, that is

$$L(\xi) = \left[\sum \gamma_i(\xi)\, y_i^-, \sum \gamma_i(\xi)\, y_i^+\right].$$

We may also compute the width of w of L in D_0 . We have

$$\begin{aligned} w(L(\xi)) &= \sum |\gamma_i(\xi)|\, w(Y_i) \\ &= \sum \gamma_i(\xi) \left(y_i^+ - y_i^-\right) \\ &= \sum \gamma_i(\xi)\, y_i^+ - \sum \gamma_i(\xi)\, y_i^-. \end{aligned}$$

Let us compute the width of L on the real line under the assumption that the intervals Y_i have a constant width W. We have

$$\begin{aligned} w(L(\xi)) &= \sum |\gamma_i(\xi)| w(Y_i) = W \sum |\gamma_i(\xi)| \\ &= W \sum |(1/S_{xx})(x_i - \bar{x})(\xi - \bar{x}) + 1/N| \\ &\leq W(1/S_{xx})|\xi - \bar{x}| \ (1 + \sum |x_i - \bar{x}|) \\ &= W + W(1/S_{xx})|\xi - \bar{x}| \sum |x_i - \bar{x}|. \end{aligned}$$

From the above formula we see that the equality is obtained in the interval D_0 . Indeed, in D_0 all γ are nonnegative and

$$\sum |\gamma_i(\xi)| = \sum \gamma_i(\xi) = \sum (x_i - \bar{x})(\xi - \bar{x})/S_{xx} + 1/N = \sum 1/N = 1$$

since $\sum (x_i - \bar{x}) = 0$. Also, it is easy to be seen that the width of L increases as we move ξ away from $\bar{x}$.

For the midpoint μ of L we have

$$\mu(L(x, Y; \xi)) = \sum \gamma_i(x; \xi) \mu(Y_i)$$

showing that the midpoint always lies on the regression line generated by the midpoint of the interval observations Y_i .

Let us compute the slope of $L(\xi)$ in the most outer intervals D_j and D_{j+1} . In D_j we have

$$\begin{aligned} L(\xi) &= \left[\sum \gamma_i(\xi) y_i^{-\text{sign}\gamma_i(\xi)}, \sum \gamma_i(\xi) y_i^{+\text{sign}\gamma_i(\xi)}\right] \\ &= \left[\sum \gamma_i(\xi) y_i^{-\text{sign}(i-j-1/2)}, \sum \gamma_i(\xi) y_i^{+\text{sign}(i-j-1/2)}\right] \end{aligned}$$

Replacing the expression for γ_i we obtain that the slope of $L(\xi)$ in D_j is

$$\left[\sum (1/S_{xx})(x_i - \bar{x}) y_i^{-\text{sign}(i-j-1/2)}, \sum (1/S_{xx})(x_i - \bar{x}) y_i^{+\text{sign}(i-j-1/2)}\right].$$

For the interval D_{j+1} we obtain the same expression.

On the other side, the interval line

$$\hat{L}(\xi) = A(x, Y)(\xi - \bar{x}) + \bar{Y}$$

has slope

$$\begin{aligned} A(x, Y) &= \{a(x, y) : y \in Y\} \\ &= \{\sum (1/S_{xx})(x_i - \bar{x}) y_i : y_i \in Y_i\} \\ &= \sum (1/S_{xx})(x_i - \bar{x}) Y_i \\ &= \left[\sum (1/S_{xx})(x_i - \bar{x}) y_i^{-\text{sign}(x_i - \bar{x})}, \sum (1/S_{xx})(x_i - \bar{x}) y_i^{-\text{sign}(x_i - \bar{x})}\right] \end{aligned}$$

showing that it coinsides with the slope of L in the most outer intervals. Taking into account that both L and $\hat{L}$ contain the segment $(\bar{x}, \bar{Y})$ we obtain sufficient information about the geometric disposition of $\bar{L}$ with respect to L.

In what follows, we shall formulate some problems that might be of certain practical interest when curve fitting is considered.

Problem 1. We saw that the interval linear function $\hat{L}(\xi) = A(x, Y)(\xi - \bar{x}) + \bar{Y}$ presents an outer approximation of the set L. However, from a practical point of view it is more interesting to find interval estimations A_1 and Y_1 for a and $\bar{y}$ so that the interval function $A_1(\xi - \bar{x}) + Y_1$ presents an inner approximation of L in certain interval for ξ. There might be considered different criteria for such an approximation.

Problem 2. Let the input vector interval $Y = Y(t)$ depend on some parameter $t \geq 0$, in such a way that for its midpoint we have $\mu(Y(t)) = \text{const}$ and its width $w(Y(t))$ is an increasing function on t. A simple such vector interval function is, e.g. the function $Y(t) = Y(0) + [-t, t], t \geq 0; Y(0)$ can be, in particular a degenerate interval vector, that is $\mu(Y(0)) = 0$. It seems to be a problem of practical interest to find the smallest t such that the set $L(t)$ generated by $Y(t)$ has a nonempty intersection with the intervals $Y(t)$. Also we may ask for the smallest t such that $L(t)$ contains a (linear) function interpolating the intervals Y_i (that is passing through the intervals).

We see that the approximation problem under interval data can be considered in relation to some additional "interpolational" requirements. Before going further into such relations we shall first recall the well known interpolation problem (under interval observations).

Interval interpolation. The simplest problem of polynomial interpolation in the numeric (noninterval) situation says that any $N+1$ input points $(x_i, y_i), i = 0, 1, \cdots, N$, in the euclidean plane R^2 generate an interpolating polynomial $p(\xi)$ of the n-th degree, which can be written, say in the form of Lagrange as

$$p(\xi) = \sum_{i=0}^{N} l_i(\xi)\, y_i,\ l_i(\xi) = \Pi_{j \neq i}(\xi - x_j) / \Pi_{j \neq i}(x_i - x_j).$$

Denote for brevity $(x_0, x_1, \cdots, x_N) = x$, and $(y_0, y_1, \cdots, y_N) = y$. Since $p(\xi)$ depends on the vectors x, y, we shall also denote the polynomial p by $p(x, y)$ and its value $p(\xi)$ at ξ by $p(x, y; \xi)$. We next consider the situation when we are given interval bounds Y_i for $y_i, i = 0, 1, \cdots, N$, instead of real numbers.

So, let us assume that we are given $N+1$ input numbers $(x_0, x_1, \cdots, x_N) = x$ and, instead of the numbers $(y_0, y_1, \cdots, y_N)$ we are now given $N+1$ intervals $(Y_1, Y_2, \cdots, Y_N) = Y$. Consider the family $\mathcal{P}$ of all interpolating polynomials $p(x, y; \cdot)$, whenever $y = (Y_0, Y_1, \cdots, Y_N)$ varies in the interval vector $Y = (Y_1, Y_2, \cdots, Y_N)$, that is the set $\mathcal{P} = \mathcal{P}(x, Y; \cdot) = \{p(x, y; \cdot) : y \varepsilon Y\}$.

The set $\mathcal{P}$ defines a set-valued function P from R to the power set of R , such that $P(\xi^*) = P(x, Y; \xi^*) = \{p(x, y; \xi^*) : y \in Y\}$.

Consider the well known

PROBLEM B. Compute the set-valued function P.

SOLUTION. Denote by $y_i^- \leq y_i^+$ the end-points of the intervals Y_i so that $Y_i = \left[y_i^-, y_i^+\right]$. We have for a fixed ξ

$$P(\xi) = \{p(x, Y; \xi) : y \varepsilon Y\} = \{\sum l_i(\xi)\, y_i : y \varepsilon Y\}$$

$$= \left[\sum l_i(\xi)\, y_i^{-\operatorname{sgn} l_i(\xi)}, \sum l_i(\xi)\, y_i^{\operatorname{sgn} l_i(\xi)}\right]$$

where

$$l_i(\xi) = \Pi_{j \neq i}(\xi - x_j) / \Pi_{j \neq i}(x_i - x_j)$$

do not depend on y and $\operatorname{sgn} l_i(\xi)$ means "+" , if $l_i(\xi) \geq 0$, and "−", if $l_i(\xi) < 0$; also $y_i^- \underset{=}{Df} y_i^+$ (right end-point) and $y_i^+ \underset{=}{Df} y_i^-$ (left end-point). This shows that $P(\xi)$ is an interval of the form $P(\xi) = [p^-(\xi),\, p^+(\xi)]$, that is, P is an interval function. Using interval arithmetic $P(\xi)$ can be written in the simple form $P(\xi) = \sum l_i(\xi) Y_i$.

The end-point functions $p^-(x, Y; \cdot)$ and $p^+(x, Y; \cdot)$ with the property

$$p^-(x, Y; \xi) = \inf_{y \in Y} p(x, y; \xi),\; p^+(x,\, Y; \xi) = \sup_{y \in Y} p(x, Y; \xi)$$

for any $\xi \in R$ present the boundaries of P, that is $P(\xi) = [p^-(x, Y; \xi),\, p^+(x, Y; \xi)]$.

Denote as before $\mu([a, b]) = (a + b)/2$, and $w([a, b]) = b - a$. Consider again an interval vector function $Y(t) = (Y_0(t), Y_1(t), \cdots, Y_N(t))$, defined for $t \varepsilon [0, T]$, such that $\mu(Y(t)) =$ const and $w(Y(t))$ is an increasing function on t , such that $w(Y(0)) = 0$. A simple example of such an interval vector function is a function of the form $Y(t) = y + [-t, t]$, for which we have $w(Y(t)) = 2t$. It seems to be of practical interest to consider the following problems.

Problem 3. Let the single-valued polynomial $p(x, Y(0))$ is a polynomial of the N-th degree and is not a polynomial of $(N - 1)$ -st degree. What is the smallest t such that the family of polynomials $p(x, Y(t))$ contains a single-valued polynomial p^* of degree less than N ? What is the approximation by p^* to the single-valued vector $(x, \mu(Y(0)))$?

A generalization of this formulation can be considered for interval observations whose centers are not fixed.

Problem 4. Let the set $p(x, Y(t))$ contain a polynomial of $(n - 1)$ -st degree. Find the largest $\tau < t$ such that $p(x, Y(\tau))$ does not contain polynomials of $(n - 1)$ -st degree.

Problem 5. Given the set of interpolating polynomials of N -th degree $P = P(x,Y) = \{p(x,y) : y \varepsilon Y\}$ for $x = (x_0, x_1, \cdots, x_N)$ and $Y = (Y_1, Y_2, \cdots, Y_N)$, find the subset of all interpolating polynomials of degree K that belong to $P, K = N-1,\ N-2, \cdots$.

6 Interval Mathematical Problems. Boundary value problems for partial differential equations involving interval parameters

As another example of interval problems we shall briefly consider boundary value problems (BVP) for partial differential equations (PDE) involving differential operators of monotone type.

Let $\mathcal{F}$ be a given class of sufficiently smooth functions defined on the set $B \subset \Re^m$. Consider the differential operator $T : \mathcal{F} \longrightarrow \Re$ and the problem

$$Tu = 0, u \in \mathcal{F}. \tag{11}$$

Definition. We say that the differential operator T is of *monotone type* if

$$Tu(x) \geq Tv(x),\ x \in B,$$

imply

$$u(x) \geq v(x),\ x \in B,$$

We shall make use of the following idea.

Let T be of monotone type and there exist functions $v(a,x)$ and $w(b,x)$, where $a, b \in \Re^k$ are parameters, such that

$$-\delta \leq Tv(a,x) \leq 0 \leq Tw(b,x) \leq \delta.$$

Then we have

$$v(a,x) \leq u(x) \leq w(b,x),\ 0 \leq w(b,x) - v(a,x) \leq c\delta,$$

where $u(x)$ is the solution of (11). Now the parameters a and b must be chosen in such a way that

$$|c\delta| = |c(a,b)\,\delta(a,b)| < \varepsilon,$$

where ε is the prescribed error bound.

Methods that make use of the above idea will be referred as δ -methods.

Let $\phi_i : B \longrightarrow \Re$, $i = 1, \cdots, n$, be a set of basic functions in the space $\mathcal{F}$. These functions are chosen to have a compact support in the domain B and such that at any point $x \in B$ only a few values of $\phi_i(x), i = 1, \cdots, n$ are distinct from zero. Usually the basis consists from finite elements or B-spline functions.

The approximate solutions $v(x)$ and $w(x)$ are represented in the form:

$$v(x) = \sum_{i=1}^{n} a_i \phi_i(x), \; w(x) = \sum_{i=1}^{n} b_i \phi_i(x), \; x \in B,$$

and the coefficients a_i and b_i are such that

$$v(x) \leq u(x) \leq w(x), \; 0 \leq w(x) - v(x) < \varepsilon, \; x \in B.$$

There are various methods for choosing the functions $\{\phi_i\}, v$ and w : finite element methods, Ritz method, Galerkin method, collocation method etc. We consider collocation δ-method and Galerkin δ-method . The domain B is divided into finite elements and the functions ϕ_i are polynomials in one element and zero outside that element.

Collocation δ -method. The upper and lower approximations w and v of the solution u are determined by the equations:

$$Tv(x_i) = -\delta, \; Tw(x_i) = +\delta, \; i = 1, \cdots, n.$$

where $\{x_i\}$ are mesh points in the domain B , the mesh being of size h. The size h is chosen sufficiently small to satisfy the inequalities:

$$\begin{cases} v(x) \leq u(x) \leq w(x), \\ 0 \leq w(x) - v(x) < \varepsilon, \; x \in B. \end{cases}$$

The parameter δ depends on the domain B, the operator T and the size of the mesh h. The next theorem gives the existence of the parameter δ.

Theorem 1. *Let T, B and h are given. Then there exists a positive number $\delta > 0$, such that : the solutions v and w of (3) corresponding to δ satisfy the inequalities*

$$\begin{cases} -2\delta \leq Tv(x) \leq 0 \leq Tw(x) \leq 2\delta, \\ 0 \leq w(x) - v(x) \leq c\delta, \text{ for every } x \in B. \end{cases}$$

The constant c depends only on the domain B and the operator T and does not depend on h . The parameter δ depends on h and $\delta \longrightarrow 0$ when $h \longrightarrow 0$.

So we can obtain lower and upper approximations of the solution $u(x)$.

More details on the δ-methods can be found in [10-13]. There exits already vast literature on the validated solution of differential equations, both PDE's and ODE's. More details on numerical methods with result verification for ODE's can be found in [1], [2], [6], [7], [17], [34], [35-37].

We hope that the solution of the above and other similar interval problems will contribute to a more effective rigorous evaluation of the effects of the imperfect modelling and to a mathematically clean motivation for the rejection of an incorrect mathematical hypothesis in a particular modelling situation.

For practical purposes in any particular situation the formulated problems can be solved by means of a program system. Such a system is now in development under the name MODYNA within the frames of a contracted study agreement between the Bulgarian Academy of Sciences and the International Institute for Applied System Analysis.

7 On the new methodology of scientific computation and its implementation.

Advanced computer arithmetic and interval analysis are suitable fundamentals for the construction of highly accurate and reliable numerical algorithms. The methodology of creating such algorithms is comprehensively described in [21]. Basically it consists of:

i) using interval analysis for delivering guarantees of the computational results and automatic validation (proof) of the inclusion of the ideal results in the computed intervals;

ii) using iterative residual (defect) correction processes in combination with the optimal computer operations (and especially the optimal scalar product) for delivering high accuracy of the final results.

On the basis of the advanced computer arithmetic and interval analysis a subroutine library for HIghly efFicient and accurate COMPutations (HIFICOMP) was developed by the Research Group for Computer Arithmetic and Interval Analysis at the Center for Informatics and Computer Technology at the Bulgarian Academy of Sciences [14], [15]. It contains subroutines for interval and computer arithmetic, for evaluation of rational expressions, for solving linear algebraic problems, differential equations, etc.

Besides the high accuracy and guarantees for the final results and the possibility of solving numerical problems with interval input data, the HIFICOMP subroutine library deliberates the scientific computer user from the final responsibility for the quality of the computational

results. The intervals produced by the HIFICOMP subroutines contain the ideal solutions with absolute guarantee and maximum accuracy. The computer user may have no doubts in the correctness of the results, he does not need to apply intuitive techniques for checking this correctness (computing remainders, repeating computations with slightly changed data or in various precisions etc.). This greatly increases the efficiency of the numerical computations from the users point of view, since he does not need to invest any time in checking the correctness of the results.

Let us mention some program systems supporting or partially supporting the new methodology for scientific computing. One of the first such programming tools was the subroutine library RINA [31], developed during the period 1982-1983 within the frames of a contract between NPL "PROGRAMA" at the Centre for Mathematics and Mechanics and the State Committee for Research and Technologies of the Bulgarian Academy of Sciences. In 1983, IBM announced the ACRITH Subroutine Library [16], developed in West Germany. A commercial version of a new programming language supporting the new approach for scientific computing appeared in 1987 - the language PASCAL-SC [18]. Other leading computer manufacturers (e.g. Siemens and Nixdorf) also offered program libraries similar to ACRITH (like ARITHMOS).

In 1987, the program library HIFICOMP (subroutine library for *hi*ghly e*ffi*cient *comp*utations) was developed as a result of a contract between the Centre for Informatics and Computer Technology and the State Committee for Research and Technologies. The library has as its goal the performance of highly accurate and safe numerical computations on computers of the IBM 370 series of compatible. The HIFICOMP subroutines compute guaranteed interval bounds for the true results of the corresponding arithmetic operations or mathematical problems to be solved. These intervals are very sharp; their endpoints are two adjacent (neighboring) machine numbers (or the intervals are even much sharper if the STC-format version of the corresponding algorithm is run).

HIFICOMP makes use of some nonstandard software-hardware tools that support the new methodology. Special tools that extend the set of arithmetic instructions available on IBM 370 and on the vector processor ES 2706 have been developed under the requirement for an easy transfer of the library in various hardware environments. To this end the dependence of the library on a particular hardware was reduced to a small number of basic subroutines written directly in some assembler type language.

This allows the subroutine library to be easily adopted for various hardware environments. The basic routines for the arithmetic operations (including the dot product) make use of a small number of machine dependable modules. Such modules are developed for IBM 370 mainframe and for personal computer IBM-PC XT/AT (so that a PC version of the HIFICOMP subroutine

library is also available). The basic routines are also microprogrammed for the vector processor EC 2706, which provides a very fast performance of the computer arithmetic. However, we should note that the software or even firmware implementation of the computer arithmetic are rather slow in comparison with what might be expected from a hardware implementation. Because of this reason a specialized processor is now in development, which will provide for a very fast execution of the whole computational process.

Examples. As an example consider the solution of the system $Ax = b$, wherein

$$A = \begin{pmatrix} 135 & 188 & 191 & 178 \\ 188 & 262 & 265 & 247 \\ 191 & 265 & 281 & 266 \\ 178 & 247 & 266 & 255 \end{pmatrix}, \quad b = \begin{pmatrix} 3516 \\ 4887 \\ 5105 \\ 4818 \end{pmatrix}.$$

The true solution is (4, 5, 6, 5). A standard program for solving linear systems, using double precision fails completely - we obtained a result reading $(-8.471\cdots, 4.592\cdots, 4.492\cdots, 5.339\cdots)$. The HIFICOMP routine LIN in STC-format with two components produces the following interval result (in single precision !):

$$X = \begin{pmatrix} [3.999999,\ 4.000001] \\ [4.999999,\ 5.000001] \\ [5.999999,\ 6.000001] \\ [4.999999,\ 5.000001] \end{pmatrix}.$$

As another example consider the linear system $Ax = b$ with

$$A = \begin{pmatrix} 19 & 30 & 11 \\ 21 & 29 & 10 \\ 17 & 31 & 12 \end{pmatrix}, \quad b = \begin{pmatrix} 60 \\ 60 \\ 60 \end{pmatrix}.$$

In this example, the matrix is singular and therefore the problem has no solution. The subroutine LIN produces a warning message "singular matrix" , whereas a "traditional" subroutine gives the following result :

$$x = (1.2733,\ 0.41007,\ 2.13666\cdots),$$

which is completely misleading.

The subroutine package MODYNA. This PC program package is under development within an IIASA contracted study agreement entitled "Mathematical Modelling of Dynamical Processes."

It will extend the arithmetical facilities of the PASCAL-SC language by means of a dynamic precision arithmetic. It will implement all features of the above outlined methodology and will contain subroutines for various interval problems arising in the course of mathematical modelling of dynamical processes. The program package MODYNA is developed by the Division on Mathematical Modelling at the Bulgarian Academy of Sciences, Institute of Biophysics, in collaboration with the Mathematical Institute and the Coordinating Centre for Informatics and Computer Technology. The main contributors are R. Angelov, P. Bochev, G. Grozev, N. Dimitrova, N. Kjurkchiev, M. Krastanov, S. Markov, V. Njagolova, K. Petrov, P. Petrov, and E. Popova.

Acknowledgements. The present research is partially supported by the Committee of Science according to contract No. 755/1988 and by IIASA in the frames of a contracted study agreement under the title "Mathematical Modelling of Dynamical Processes."

8 References

1. Angelov R. and S. Markov. Two-sided approximation of the solution of the initial problem for systems of ordinary differential equations involving inexact data. Constructive Theory of Functions '84. Publ. House of the Bulg. Acad. of Sciences. Sofia. 1984. 125-127.

2. Alefeld, G. and J. Herzberger. Introduction to interval computations. Academic Press. New York. 1981.

3. Angelov, R., P. Bochev, G. Grozev, and S. Markov. Highly accurate and safe numerical computations via interval analysis and advanced computer arithmetic. International Conference on Numerical Methods and Applications. Sofia. 22-27 August, 1988.

4. Auzinger, W. and H.J. Stetter. Accurate arithmetic results for decimal data on non-decimal computers. *Computing* 35, 1985.

5. Behler, J.H., U.W. Kulisch, M. Metzger, S.M. Rump, Ch. Ullrich, and W. Walter. FORTRAN-SC: A study of FORTRAN Extension for Engineering/Scientific Computation with access to ACRITH. *Computing* 39. 1987. 93-110.

6. Bochev, P. and S. Markov. A self-validating numerical method for the matrix exponential. *Computing,* 43, 59-72 1989.

7. Boehmer, K., P. Hemker and H.J. Stetter. The defect correction approach, in: Defect Correction Methods; Theory and Applications. Computing Supplementum. Springer. Wien. 1984.

8. Bohlender, G. What do we need beyond IEEE Arithmetic? In: Computer arithmetic and self-validating numerical methods. (Ed. Ch. Ulrich) Academic Press, 1990, 1-32.

9. Dimitrova, N. and S. Markov. Interval-arithmetic algorithms for simultaneous computation of all polynomial zeros. In: Contributions to computer arithmetic and self-validating numerica methods. Ch. Ullrich (Ed.) IMACS Annals on computing and applied mathematics, vol. 7 (1990), J.C. Baltzer A.G. Aci. publ. co., Basel, Switzerland.

10. Grozev, G. One-sided algorithms for boundary value problems. Mathematics and Education in Mathematics, 1988. Proc. 17th Spring Conf. of the Union of Bulgarian Mathematicians, April 1988. Publ. House of the Bulgarian Academy of Sciences. 427-431.

11. Grozev, G. Two-sided difference methods for approximation of the solution of the parabolic partial differential equations. Mathematics and Education in Mathematics 1984. BAN Sofia.

12. Grozev, G. and S. Markov. An interval method for a two-point boundary value problems using cubic splines. Collect. of scientific papers honouring Prof. K. Nickel on occasion of his 60-th birthday. Part 1. (ed. by J. Garloff) Inst. f. Angew. Math. Univ. Freiburg. i. Br. 1984.

13. Grozer, G. and S. Markov. Numerical Methods with Verification for Boundary Value Problems. In: Contributions to computer arithmetich and self-validating numerical methods. Ch. Ullrich (Ed.) IMACS Annals on computing and applied mathematics, vol. 7 (1990), J.C. Baltzer A.G. Sci. publ. co., Basel, Switzerland.

14. HIFICOMP. Subroutine Library for *hi*ghly ef*fi*cient and accurate *comp*utations. Program Description and User's Guide, CINTI Registration Number 1.a.006.02112-01 13 (1987).

15. HIFICOMP. Subroutine Library for *hi*ghly ef*fi*cient and accurate *comp* utations, Methodological guide. Center for Informatics and Computer Technology. Bulgarian Academy of Sciences, 1987, CINTI Reg.No.1.A.066.02112-01 37 (1987).

16. IBM High-accuracy Arithmetic Subroutine Library (ACRITH) Program Description and User's Guide, SC 33-6164-02 3rd Edition. April 1986.

17. Kaucher, E. and W.L. Miranker. Self-validating numerics for function space problems. Academic Press. New York. 1984.

18. Kulisch, U.W. (ed.). PASCAL-SC: A PASCAL extension for Scientific Computations; Information manual and floppy disks; version IBM PC/AT; operating system DOS. B.G.

Teubner Verlag (Willey-Teubner series in computer science). Stuttgart. 1987.

19. Kulisch, U.W., and W.L. Miranker. Computer Arithmetic in Theory and Practice. Academic Press. New York. 1981.

20. Kulisch, U.W., and W.L. Miranker. A new approach to scientific computation. Academic Press. New York. 1983.

21. Kulisch, U.W., and W.L. Miranker. The arithmetic of the digital computer; a new approach. SIAM Review. Vol.28, No 1. March 1986. 1-40.

22. Kulisch, U. and H.J. Stetter (eds). Automatic Result Verification. Computing, Suppl. 6. 1988. 1-6.

23. Markov, S. Some applications of extended interval arithmetic to interval iterations. Computing, Suppl. 2. 1980. 69-84.

24. Markov, S. Extended interval arithmetic: Part One. Freiburger Intervall-Berichte, 80/8. 1-40. 1980.

25. Markov, S. Calculus for interval functions of a real variable. *Computing* 22. 325-337. 1979.

26. Markov, S. On the numerical algorithms formulated in computer arithmetic. Proc. of the 6th Symposium on computer arithmetic, Aarchus, Denmark, 1983. IEEE Computer Society Press, 122-128.

27. Markov, S. Mathematical fundamentals of numerical computation. Mathematics and Education in Mathematics, 1988. Proc. 17th Spring conf. of the Union of Bulgarian Mathematicians, April, 1988. Publ. House of the Bulgarian Academy of Sciences, 83-90.

28. Moore, R. Interval analysis. Prentice-Hall. Englewood Cliffs, N.J. 1966.

29. Moore, R. Methods and applications of interval analysis, SIAM, Philadelphia, 1979.

30. Nickel, K. Interval analysis, in: "The state of art in numerical analysis". Edited by D. Jacobs. Academic Press. New York. 1977.

31. RINA. Subroutine Library for Reliable Interval Numerical Algorithms. CINTI Registration Number (1983), NPL "PROGRAMA".

32. Rump, S.M. Solution of linear and nonlinear algebraic problems with sharp, guaranteed bounds. Computer Suppl. 5. 1984.

33. Sendov, Bl. Practical mathematical analysis. Proc. of the 6th Spring Conference on Mathematics and Education in Mathematics of the Union of Bulgarian Mathematicians. April 6-9, 1977. Varna. Publ. House of the Bulg. Acad. of Sciences. Sofia. 1977. 1-40.

34. Shokin, Yu. I., S.A. Kalmykov, and Z.H. Yuldashev. Methods of interval analysis. Nauka. Novosibirsk. 1986 (in Russian).

35. Stetter, H.J. Sequential defect correction for high accuracy floating-point algorithms, in: Numerical Analysis (Proceedings, Dundee 1983). Lecture Notes in Math., vol. 1066.

36. Stetter, H.J. Inclusion algorithms with functions as data. Computing, Suppl. 6. 1988. 213-224.

37. Stetter, H.J. Validated Solution of Initial Value Problems for ODE. In: Computer arithmetic and self-validating numerical methods. C. Ullrich (Ed.) Academic Press, 1990, 171-187.

38. Stoyanova, B. and R. Angelov. Combining the approach of safe and accurate computation with the approach of parallel computing. Mathematics and Education in Mathematics, 1988. Proc. 17th Spring Conf. of the Union of Bulgarian Mathematicians, April 1988. Publ. House of the Bulgarian Academy of Sciences, 496-500.

39. Ullrich, C. (Ed.) Computer arithmetic and Self-validating numerical methods. Academic Press, 1990.

40. Ullrich, C. (Ed.) Contributions to computer arithmetic and self-validating numerical methods. IMAC Annals on Computing and applied mathematics, vol. 7 1990, J.C. Baltzer A.G. Sci. publ. co., Basel, Switzerland.

41. Wilkinson, J.H. Rounding errors in algebraic processes. Prentice-Hall, Englewood Cliffs, N.J. 1963.

42. Yohe, J.M. Rounding in floating-point arithmetic. IEEE Trans. Comput. C-22. 577-586. 1973.