Testing Implementations of PPS-methods for Interval Linear Systems*

Dmitri Yu. Lyudvin

Institute of Computational Technologies SB RAS,
Novosibirsk, Russia
lyudvin@ngs.ru

Sergey P. Shary

Institute of Computational Technologies SB RAS and
Novosibirsk State University, Novosibirsk, Russia
shary@ict.nsc.ru

Abstract

This paper compares various implementations of the parameter partitioning methods (PPS-methods) for optimal outer estimation of the solution sets to interval linear systems. Special attention is focused on the analysis of the modification based on Rohn's technique, which is the most complex, laborious, but the most efficient one. We present results of numerical experiments with several computational schemes as well as their analysis, and we formulate practical recommendations on how to optimize the computational schemes of the parameter partitioning methods.

Keywords: interval linear equations, solution set, optimal enclosure, PPS-method, parameter partitioning methods, Rohn theorem, sequential guarantee, final guarantee.

AMS subject classifications: 65G40, 65F10

1 Introduction

We consider the problem of computing the tightest (optimal) component-wise bounds on the solution set to an interval linear system of the form

$$\left\{\begin{array}{ccccccccc} \boldsymbol{a}_{11}x_1 & + & \boldsymbol{a}_{12}x_2 & + & \ldots & + & \boldsymbol{a}_{1n}x_n & = & \boldsymbol{b}_1, \\ \boldsymbol{a}_{21}x_1 & + & \boldsymbol{a}_{22}x_2 & + & \ldots & + & \boldsymbol{a}_{2n}x_n & = & \boldsymbol{b}_2, \\ \vdots & & \vdots & & \ddots & & \vdots & & \vdots \\ \boldsymbol{a}_{n1}x_1 & + & \boldsymbol{a}_{n2}x_2 & + & \ldots & + & \boldsymbol{a}_{nn}x_n & = & \boldsymbol{b}_n, \end{array}\right. \tag{1}$$

*Submitted: December 23, 2012; Revised: September 17, 2013; Accepted: December 16, 2013.

or, briefly,

$$\boldsymbol{A}x = \boldsymbol{b}, \tag{2}$$

with an interval $n \times n$-matrix $\boldsymbol{A}$ and an interval n-vector $\boldsymbol{b}$. The interval linear system (1)–(2) is understood as a family of point linear systems of the same structure, with matrices and right-hand side vectors within the interval matrix $\boldsymbol{A}$ and the interval vector $\boldsymbol{b}$, respectively.

The united solution set of the interval linear system is the set

$$\Xi(\boldsymbol{A}, \boldsymbol{b}) \;=\; \{\, x \in \mathbb{R}^n \mid (\exists\, A \in \boldsymbol{A})(\exists\, b \in \boldsymbol{b})(\, Ax = b\,) \,\},$$

formed by solutions to all the point systems $Ax = b$ with $A \in \boldsymbol{A}$ and $b \in \boldsymbol{b}$. There are many other definitions for solution sets to interval systems of equations, but in this paper, we confine ourselves to the united solution set $\Xi(\boldsymbol{A}, \boldsymbol{b})$. Hence, we call it just the *solution set*.

We suppose that the interval matrix $\boldsymbol{A}$ is regular (nonsingular) in the system (1)–(2), that is, $\boldsymbol{A}$ contains only regular point matrices. Hence, the solution set $\Xi(\boldsymbol{A}, \boldsymbol{b})$ of the interval linear system $\boldsymbol{A}x = \boldsymbol{b}$ is bounded.

The problem of optimal outer estimation of the solution set to the interval linear systems (1)–(2) can be formulated as follows:

find an interval box $\boldsymbol{U}$ that has the smallest possible width and contains the solution set $\Xi(\boldsymbol{A}, \boldsymbol{b})$ of the interval linear system

, (3)

or, in component-wise form,

find $\min\{\, x_\nu \mid x \in \Xi(\boldsymbol{A}, \boldsymbol{b})\}$ and $\max\{\, x_\nu \mid x \in \Xi(\boldsymbol{A}, \boldsymbol{b})\}$, $\nu = 1, 2, \ldots, n$, or their most precise estimates from below and from above, respectively

. (4)

In our paper, we adhere to the second formulation and confine ourselves to computing only $\min\{\, x_\nu \mid x \in \Xi(\boldsymbol{A}, \boldsymbol{b})\}$, since the following equality holds for any fixed ν:

$$\max\{\, x_\nu \mid x \in \Xi(\boldsymbol{A}, \boldsymbol{b})\} = -\min\{\, x_\nu \mid x \in \Xi(\boldsymbol{A}, -\boldsymbol{b})\}.$$

Note that the problem of interval estimation of the solution set to interval linear systems is NP-hard [9].

We consider a class of efficient numerical algorithms for the solution of the outer estimation problem (3) which is called the class of "parameter partitioning methods", or PPS-methods, developed in [20, 23]. PPS-methods sequentially refine outer component-wise estimates of the solution set to the interval linear system $\boldsymbol{A}x = \boldsymbol{b}$ by subdividing interval elements in the matrix $\boldsymbol{A}$ and right-hand side vector $\boldsymbol{b}$.

The subdivision of the interval system can be simplified crucially by applying the *Beeck-Nickel Theorem* [2, 12]: if an interval matrix $\boldsymbol{A}$ is regular, then for any $\nu \in \{1, 2, \ldots, n\}$, exact component-wise estimates of the points from the solution set are attained at the endpoint matrices and right hand-side vectors, made up of endpoints of the interval elements of $\boldsymbol{A}$ and $\boldsymbol{b}$.

Beeck-Nickel Theorem has been strengthened by J. Rohn [16] who revealed that, if the matrix $\boldsymbol{A}$ is regular, then both minimal and maximal component-wise values of the points from the solution set are attained at the set of no more than 2^n extreme

solutions to the equation $|(\mathrm{mid}\,\boldsymbol{A})x - \mathrm{mid}\,\boldsymbol{b}| = \boldsymbol{A}\cdot|x| + \mathrm{rad}\,\boldsymbol{b}$. This fact can be used for further modification of the subdivision process in the parameter partitioning methods.

The purpose of the present work is to test and compare various implementations of PPS-methods that use

1) Rohn's technique for eliminating unpromising endpoint combinations;
2) a monotonicity test with respect to the elements of the matrix and right-hand side vector of the system;
3) various enclosure methods for interval linear systems;
4) various ways of processing the results of the partition of the initial interval linear system are stored.

Results of the numerical experiments with several computational schemes as well as their analysis and conclusions are presented below.

2 PPS-method for Interval Linear Systems and its Modifications

The parameter partitioning methods consist of sequential refinement of the estimate $\min\{\, x_\nu \mid x \in \Xi(\boldsymbol{A},\boldsymbol{b})\}$ by subdividing the initial system $\boldsymbol{A}x = \boldsymbol{b}$ into "systems-descendants", through breaking up an interval element of either the matrix $\boldsymbol{A}$ or the vector $\boldsymbol{b}$ to its endpoints. Below, we give a short survey of PPS-methods. Their thorough consideration can be found in [20, 22, 23].

We will use the following notation:

$\boldsymbol{A}'$, $\boldsymbol{A}''$ are matrices obtained from $\boldsymbol{A}$ by replacing a certain interval element by its lower and upper endpoints, respectively;

$\boldsymbol{b}'$, $\boldsymbol{b}''$ are vectors obtained from $\boldsymbol{b}$ by replacing a certain interval component by its lower and upper endpoints, respectively;

Encl is a fixed method that computes an enclosure of the solution set (we shall call it a *basic enclosure method*);

$\mathit{Encl}\,(\boldsymbol{Q},\boldsymbol{r}) \in \mathbb{IR}^n$ is an interval enclosure of the solution set $\Xi(\boldsymbol{Q},\boldsymbol{r})$ produced by the method *Encl*, so that $\mathit{Encl}\,(\boldsymbol{A},\boldsymbol{b}) \supseteq \Xi(\boldsymbol{A},\boldsymbol{b})$;

$\Upsilon(\boldsymbol{A},\boldsymbol{b}) = (\mathit{Encl}\,(\boldsymbol{A},\boldsymbol{b}))_\nu$ is the lower endpoint of the ν-th component, $\nu \in \overline{\{1,2,\ldots,n\}}$, of the interval enclosure $\mathit{Encl}\,(\boldsymbol{A},\boldsymbol{b})$.

If the estimate $\Upsilon(\boldsymbol{A},\boldsymbol{b})$ is inclusion monotone with respect to $\boldsymbol{A}$ and $\boldsymbol{b}$, then having solved the interval "systems-descendants" $\boldsymbol{A}'x = \boldsymbol{b}'$ and $\boldsymbol{A}''x = \boldsymbol{b}''$ obtained from $\boldsymbol{A}x = \boldsymbol{b}$, we can get a better estimate for $\min\{\, x_\nu \mid x \in \Xi(\boldsymbol{A},\boldsymbol{b})\}$ from below in the form of

$$\min\{\, \Upsilon(\boldsymbol{A}',\boldsymbol{b}'), \Upsilon(\boldsymbol{A}'',\boldsymbol{b}'')\}.$$

Then we can repeat the procedure with respect to the "systems-descendants" $\boldsymbol{A}'x = \boldsymbol{b}'$ and $\boldsymbol{A}''x = \boldsymbol{b}''$, which results in further improvement of the estimate. Then it makes sense to repeat such recalculation again and again ...

Overall, we arrange an iterative procedure for refining the estimate for $\min\{\, x_\nu \mid x \in \Xi(\boldsymbol{A},\boldsymbol{b})\}$ in accordance with the well-known "branch-and-bound" method (see the details in [20, 22, 23]). For the natural stopping of the algorithm, it is required to reach the complete deintervalization of the initial interval system or the estimates with the accuracy lower than a small positive tolerance ε.

During the execution of the algorithm, we maintain a *working list* of records in which the information about the systems-descendants is stored. This information is used substantially at further steps of the algorithm, and the algorithm is essentially adaptive. A special feature of the PPS-methods is that they generate a sequence of estimates for minimal and maximal values of the components of the solution set from below and from above, respectively. Hence, when terminated at any time, the algorithm still produces a correct solution to the outer estimation problem (3)–(4).

At every step of the algorithm, we use the basic enclosure method *Encl* that computes an enclosure of the solution set to the systems descendants. We implemented the following basic enclosure methods:

- Krawczyk method [8],
- modified Krawczyk method with epsilon inflation (see e.g. [19]),
- interval Gauss method (see e.g., [10, 22]),
- interval Gauss-Seidel method (see e.g., [10, 22]),
- Hansen-Bliek-Rohn procedure (described e.g., in [4] or Neumaier's paper [11]).

As a basic enclosure method, we also have tested the procedure `verifylss` from the popular package INTLAB [18] (see its description e.g., in [7]). The procedure has two stages. The first stage is based on the modified Krawczyk method with epsilon inflation, while the second stage uses the Hansen-Bliek-Rohn procedure. If a desired enclosure of the solution set is not found after seven iterations in the first stage, then the second stage starts.

2.1 Rohn Modification

In [16], J. Rohn proposed an approach to optimal outer estimation of the solution sets to interval linear systems based on examination of extreme solutions. He proved that, in the case of a square regular matrix $\boldsymbol{A}$, minimal and maximal values of the components of the points from the solution set $\Xi(\boldsymbol{A}, \boldsymbol{b})$ are attained at the set of no more than 2^n solutions to point systems

$$\big|(\operatorname{mid} \boldsymbol{A})x - \operatorname{mid} \boldsymbol{b}\big| = \operatorname{rad} \boldsymbol{A} \cdot |x| + \operatorname{rad} \boldsymbol{b}$$

obtained from the well-known Oettli-Prager inequality [14] by equating its left-hand and right-hand sides. Such solutions are called *extreme solutions* for the interval system $\boldsymbol{A}x = \boldsymbol{b}$. It turns out that the convex hull of all the extreme solutions coincides with that of the entire solution set $\Xi(\boldsymbol{A}, \boldsymbol{b})$.

Let us denote the set of n-vectors with the components ± 1 by $\mathcal{E}$. The set $\mathcal{E}$ has 2^n elements. For fixed vectors σ and $\tau \in \mathcal{E}$, we define the matrices T_σ and $A^{\sigma\tau} = \{a_{ij}^{\sigma\tau}\}$ and the vector $b^\sigma = \{b_i^\sigma\}$ as follows:

$$T_\sigma = \operatorname{diag}\{\sigma_1, \ldots, \sigma_n\},$$

$$a_{ij}^{\sigma\tau} = \begin{cases} \overline{\boldsymbol{a}}_{ij}, & \text{if } \sigma_i\tau_j = -1, \\ \underline{\boldsymbol{a}}_{ij}, & \text{if } \sigma_i\tau_j = 1, \end{cases} \qquad b_i^\sigma = \begin{cases} \overline{\boldsymbol{b}}_i, & \text{if } \sigma_i = 1, \\ \underline{\boldsymbol{b}}_i, & \text{if } \sigma_i = -1. \end{cases}$$

The matrix $A^{\sigma\tau}$ and the vector b^σ are made up of collections of endpoints of the elements of $\boldsymbol{A}$ and $\boldsymbol{b}$, respectively, so that the system $A^{\sigma\tau}x = b^\sigma$ is actually an "extreme point" for the given interval system $\boldsymbol{A}x = \boldsymbol{b}$.

Rohn's Theorem on Extreme Solutions [16]. *Let an $n \times n$-matrix $\boldsymbol{A}$ be regular and $\boldsymbol{b}$ be an interval n-vector. Then, for every $\sigma \in \mathcal{E}$, the equation*

$$\operatorname{mid} \boldsymbol{A} \cdot x - T_\sigma \cdot \operatorname{rad} \boldsymbol{A} \cdot |x| = b^\sigma$$

has a unique solution x^σ within $\Xi(\boldsymbol{A}, \boldsymbol{b})$ and

$$\operatorname{conv} \Xi(\boldsymbol{A}, \boldsymbol{b}) = \operatorname{conv} \{\, x^\sigma \mid \sigma \in \mathcal{E} \}$$

holds, where conv *denotes convex hull.*

In other words, the solution set $\Xi(\boldsymbol{A}, \boldsymbol{b})$ contains a finite and uniquely defined family of extreme solutions that determines the convex hull of the solution set as well as its interval hull. Then, computing all the extreme solutions and comparing them with each other, we can get optimal estimates of the solution set after a finite number of steps. An important point is that the number 2^n of the extreme solutions is significantly less than the number 2^{n^2+n} of the solutions to all the point systems with matrices and right-hand side vectors formed by the endpoints of the interval entries from $\boldsymbol{A}$ and $\boldsymbol{b}$, respectively. To put it differently, Rohn's result significantly reduces the complexity of the exhaustive search among the "endpoint" linear systems suggested by the Beeck-Nickel Theorem.

It makes sense to use Rohn's Theorem in PPS-methods. The corresponding computational scheme has been elaborated in [22, 23]. Here, we give a brief description. Given a regular matrix $\boldsymbol{A}$, the extreme component-wise values for the points from the solution set $\Xi(\boldsymbol{A}, \boldsymbol{b})$ can be reached only at the set of 4^n matrices $A^{\sigma\tau}$ and associated vectors b^σ

$$\min\{\, x_\nu \mid x \in \Xi(\boldsymbol{A}, \boldsymbol{b})\} = \min_{\sigma,\tau \in \mathcal{E}} \left((A^{\sigma\tau})^{-1} b^\sigma \right)_\nu, \qquad \nu = 1, 2, \ldots, n.$$

As the result, when partitioning the parameters of the interval system, we can look at the endpoints of the interval elements of the matrix and of the right-hand side vector to be subdivided and determine whether the endpoint combination is allowed by Rohn theorem. In doing this, we connect with every interval system $\boldsymbol{Q}x = \boldsymbol{r}$, produced at every step of the algorithm, an auxiliary integer $n \times n$-matrix $W = \{w_{ij}\}$ with the elements ± 1 or 0 and auxiliary integer n-vectors $s = \{s_i\}$ and $t = \{t_j\}$ with the elements ± 1 or 0, such that

$$w_{ij} = \begin{cases} -1, & \text{if} \quad \boldsymbol{q}_{ij} = \overline{\boldsymbol{a}}_{ij}, \\ 0, & \text{if} \quad \boldsymbol{q}_{ij} = \boldsymbol{a}_{ij}, \\ 1, & \text{if} \quad \boldsymbol{q}_{ij} = \underline{\boldsymbol{a}}_{ij}, \end{cases} \qquad s_i = \begin{cases} -1, & \text{if} \quad \boldsymbol{r}_i = \underline{\boldsymbol{b}}_i, \\ 0, & \text{if} \quad \boldsymbol{r}_i = \boldsymbol{b}_i, \\ 1, & \text{if} \quad \boldsymbol{r}_i = \overline{\boldsymbol{b}}_i, \end{cases}$$

and

$$w_{ij} = s_i t_j, \qquad i, j = 1, 2, \ldots, n.$$

The values of t_j are determined through the matrix W and the vector s. The matrix W is called the *check matrix*, and the vectors s, t are called *check vectors*. At the start of the algorithm, we set W, s, and t to all zeros. Then, after partition of the interval system during the algorithm execution, they are recalculated at every step.

Taking into account the values of the check matrix W and vector s, we perform, at each step of the algorithm, subdivision of an interval element in the *leading system* $\boldsymbol{Q}x = \boldsymbol{r}$ that provides the smallest current estimate for $\{x_\nu \mid x \in \Xi(\boldsymbol{A}, \boldsymbol{b})\}$. When subdividing an element $\boldsymbol{q}_{kl}$ of the matrix $\boldsymbol{Q}$ that corresponds to $w_{kl} = 0$, we generate

two systems descendants $\boldsymbol{Q}'x = \boldsymbol{r}'$ and $\boldsymbol{Q}''x = \boldsymbol{r}''$. If $w_{kl} = \pm 1$, we generate only one descendant, depending on the sign of w_{kl}. Similarly, we perform subdivision of an element in the right-hand side vector $\boldsymbol{r}$, depending on the values of the vector s.

After having partitioned the leading system, the check matrices and vectors for the systems descendants are calculated anew. If at least one object of the triple (W', s', t') is changed, the two remaining ones are recalculated according to the equalities $w_{ij} = s_i t_j$, $i, j = 1, 2, \ldots, n$ (see [22, 23]).

2.2 Monotonicity Test

Use of a monotonicity test within the solution of interval linear systems has been first proposed in [6], but in PPS-methods, combined with the subdivision technique, it becomes especially powerful.

Let the interval linear system $\boldsymbol{Q}x = \boldsymbol{r}$ be given. We can find the interval extensions of the derivatives

$$\frac{\partial x_\nu(Q,r)}{\partial q_{ij}}, \qquad \frac{\partial x_\nu(Q,r)}{\partial r_i}$$

of the ν-th component of the solution $x(Q,r)$ to the point system $Qx = r$ with respect to ij-th element of the matrix Q and the i-th component of the vector r by

$$\frac{\partial x_\nu(\boldsymbol{Q},\boldsymbol{r})}{\partial q_{ij}} = -\boldsymbol{y}_{\nu i}\boldsymbol{x}_j, \qquad \frac{\partial x_\nu(\boldsymbol{Q},\boldsymbol{r})}{\partial r_i} = \boldsymbol{y}_{\nu i},$$

where $\boldsymbol{Y} = (\boldsymbol{y}_{ij})$ is "inverse interval matrix" for $\boldsymbol{Q}$, i. e. $\boldsymbol{Y} \supseteq \{\, Q^{-1} \mid Q \in \boldsymbol{Q} \,\}$ (see details e.g. in [6]).

If the interval $n \times n$-matrix $\tilde{\boldsymbol{Q}}$ and the interval n-vector $\tilde{\boldsymbol{b}}$ are formed of the elements

$$\tilde{\boldsymbol{q}}_{ij} = \begin{cases} \left[\underline{\boldsymbol{q}}_{ij}, \underline{\boldsymbol{q}}_{ij}\right], & \text{if } \dfrac{\partial x_\nu(\boldsymbol{Q},\boldsymbol{r})}{\partial q_{ij}} \geq 0, \\ \left[\overline{\boldsymbol{q}}_{ij}, \overline{\boldsymbol{q}}_{ij}\right], & \text{if } \dfrac{\partial x_\nu(\boldsymbol{Q},\boldsymbol{r})}{\partial q_{ij}} \leq 0, \\ \boldsymbol{q}_{ij}, & \text{if } \operatorname{int}\dfrac{\partial x_\nu(\boldsymbol{Q},\boldsymbol{r})}{\partial q_{ij}} \ni 0, \end{cases} \qquad \tilde{\boldsymbol{r}}_i = \begin{cases} \left[\underline{\boldsymbol{r}}_i, \underline{\boldsymbol{r}}_i\right], & \text{if } \dfrac{\partial x_\nu(\boldsymbol{Q},\boldsymbol{r})}{\partial r_i} \geq 0, \\ \left[\overline{\boldsymbol{r}}_i, \overline{\boldsymbol{r}}_i\right], & \text{if } \dfrac{\partial x_\nu(\boldsymbol{Q},\boldsymbol{r})}{\partial r_i} \leq 0, \\ \boldsymbol{r}_i, & \text{if } \operatorname{int}\dfrac{\partial x_\nu(\boldsymbol{Q},\boldsymbol{r})}{\partial r_i} \ni 0, \end{cases}$$

where "int" means interior of intervals, then the following equality is evidently true

$$\min\{\, x_\nu \mid x \in \Xi(\tilde{\boldsymbol{Q}}, \tilde{\boldsymbol{r}})\} = \min\{\, x_\nu \mid x \in \Xi(\boldsymbol{Q}, \boldsymbol{r})\}.$$

Since the number of interval elements with nonzero width in $\tilde{\boldsymbol{Q}}$ and $\tilde{\boldsymbol{r}}$ can be substantially less than that in $\boldsymbol{Q}$ and $\boldsymbol{r}$, reducing the interval system $\boldsymbol{Q}x = \boldsymbol{r}$ to the system $\tilde{\boldsymbol{Q}}x = \tilde{\boldsymbol{r}}$ simplifies the computation of the desired $\min\{\, x_\nu \mid x \in \Xi(\boldsymbol{Q}, \boldsymbol{r})\}$.

To summarize, the monotonicity test [20, 22] is useful when applied before partitioning of the leading interval system, which results in deintervalization of some interval elements of its matrix. Then only such elements of the matrix $\boldsymbol{Q}$ have non-zero widths that the interval extensions of their derivatives contain zero.

2.3 Structure of the Working List and its Processing

We also analysed implementations of PPS-methods that use various structures and ways of processing the working list $\mathcal{L}$ where the results of the partitioning of the initial interval linear system are stored. We considered the following options:

1. $\mathcal{L}$ is an unordered list of records (a heap);
2. the records of $\mathcal{L}$ are in ascending order by the estimate $\Upsilon(\boldsymbol{Q}, \boldsymbol{r})$;
3. within $\mathcal{L}$, an ordered sublist $\mathcal{L}_l$ of the active records is organized that has a fixed maximal length, and the remaining records are stored as a heap;
4. Pankov's method [15], in which a threshold constant γ is defined and an ordered sublist $\mathcal{L}_\gamma$ of the active records is separated, for which $\Upsilon(\boldsymbol{Q}, \boldsymbol{r}) < \gamma$; the complement $\mathcal{L}\backslash\mathcal{L}_\gamma$ is stored as a heap.

During the algorithm run, we delete unpromising records from the working list $\mathcal{L}$ or the active records sublist, $\mathcal{L}_l$ or $\mathcal{L}_\gamma$. This procedure [20, 22] has no effect on the result of the algorithm, but it intensifies the processing of the working list.

If the active records sublist, either $\mathcal{L}_l$ or $\mathcal{L}_\gamma$, becomes empty, then a new ordered sublist is formed from the working list $\mathcal{L}$. The threshold constant γ is recalculated for the new sublist $\mathcal{L}_\gamma$.

3 Test Interval Systems

We have implemented the algorithms described in the preceding sections in the MATLAB based interval toolbox INTLAB. Various modifications of PPS-methods have been tested on the following interval systems.

Example 1. Neumaier interval linear system [10]:

$$\begin{pmatrix} \theta & [0,2] & \cdots & [0,2] \\ [0,2] & \theta & \cdots & [0,2] \\ \vdots & \vdots & \ddots & \vdots \\ [0,2] & [0,2] & \dots & \theta \end{pmatrix} \quad x = \begin{pmatrix} [-1,1] \\ [-1,1] \\ \vdots \\ [-1,1] \end{pmatrix},$$

where θ is a nonnegative real parameter. The matrix of the Neumaier system of even order n is regular for $\theta > n$, while the matrix of odd order n is regular for $\theta > \sqrt{n^2-1}$ [10]. As θ approaches the boundary of singularity, the size of the solution set increases infinitely. Varying θ, we can get a collection of model test problems for numerical experiments with interval linear systems.

Figure 1 shows the solution set to a particular case of the Neumaier system with dimension 3 and $\theta = 4$;

$$\begin{pmatrix} 4 & [0,2] & [0,2] \\ [0,2] & 4 & [0,2] \\ [0,2] & [0,2] & 4 \end{pmatrix} \quad x = \begin{pmatrix} [-1,1] \\ [-1,1] \\ [-1,1] \end{pmatrix}, \tag{5}$$

Example 2. Shary interval linear system [21, 22]:

$$\begin{pmatrix} [n-1,N] & [\alpha-1,1-\beta] & \cdots & [\alpha-1,1-\beta] \\ [\alpha-1,1-\beta] & [n-1,N] & \cdots & [\alpha-1,1-\beta] \\ \vdots & \vdots & \ddots & \vdots \\ [\alpha-1,1-\beta] & [\alpha-1,1-\beta] & \dots & [n-1,N] \end{pmatrix} \quad x = \begin{pmatrix} [1-n,n-1] \\ [1-n,n-1] \\ \vdots \\ [1-n,n-1] \end{pmatrix},$$

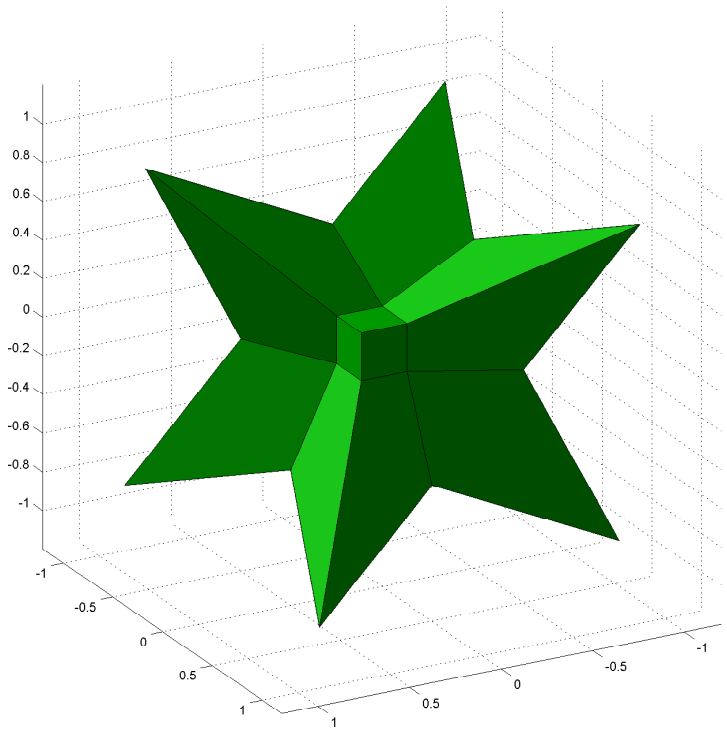

Figure 1: Solution set of the Neumaier interval system (5).

where n is the dimension of the system ($n \geq 2$), α and β are parameters satisfying $0 < \alpha \leq \beta \leq 1$, and N is such a real number that $N \geq n - 1$. As β decreases and approaches zero, the matrix of the system tends to a singular one, and its solution set $\tilde{\Xi}$ infinitely grows in size. Varying the ratio between α and β, we can modify the shape of the solution set. The tightest component-wise estimates of the solution set $\tilde{\Xi}$ are independent of N:

$$\begin{aligned} \min \{ x_i \mid x \in \tilde{\Xi} \} &= -1/\alpha, \\ \max \{ x_i \mid x \in \tilde{\Xi} \} &= 1/\alpha, \qquad i = 1, 2, \ldots, n\,. \end{aligned}$$

Figure 2 shows the solution set to the particular case of the Shary interval system corresponding to $n = 3$, $N = 4$, $\alpha = 0.15$, $\beta = 0.2$:

$$\begin{pmatrix} [2,4] & [-0.85,0.8] & [-0.85,0.8] \\ [-0.85,0.8] & [2,4] & [-0.85,0.8] \\ [-0.85,0.8] & [-0.85,0.8] & [2,4] \end{pmatrix} \qquad x \;=\; \begin{pmatrix} [-2,2] \\ [-2,2] \\ [-2,2] \end{pmatrix}. \tag{6}$$

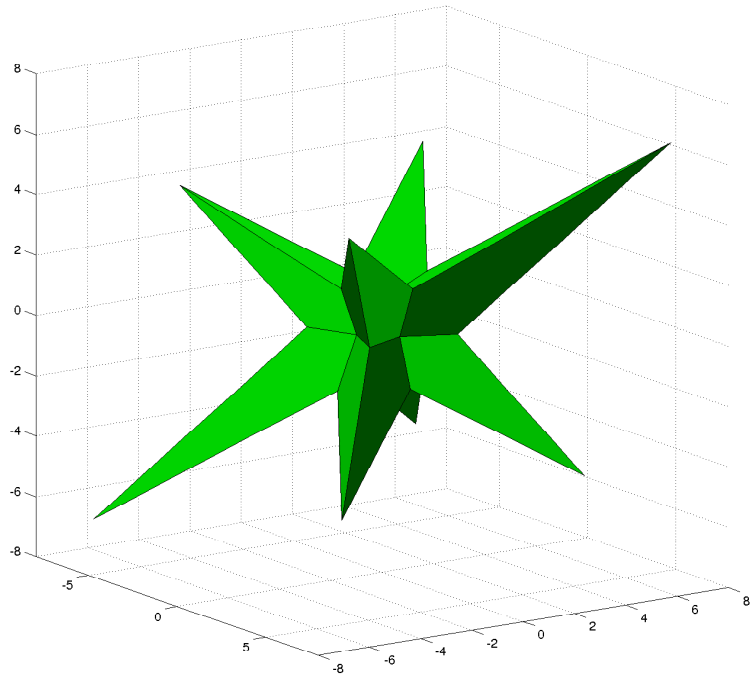

Figure 2: Solution set of the Shary interval system (6).

Example 3. The interval linear system from Toft's paper [24], with the matrix

$$\boldsymbol{A} = \begin{pmatrix} [1-r,1+r] & 0 & \cdots & [1-r,1+r] \\ 0 & [1-r,1+r] & \cdots & [2-r,2+r] \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & [n-1-r,n-1+r] \\ [1-r,1+r] & [2-r,2+r] & \cdots & [n-r,n+r] \end{pmatrix}$$

and the right-hand side vector

$$\boldsymbol{b} = \begin{pmatrix} [1-R,1+R] \\ [1-R,1+R] \\ \vdots \\ [1-R,1+R] \end{pmatrix},$$

where r and R are positive real numbers. It is an "intervalization" of a test system from the handbook [5].

Figure 3 shows the solution set to the 3-dimensional Toft interval linear system [24] that corresponds to $r = 0.1$ and $R = 0.2$:

$$\begin{pmatrix} [0.9,1.1] & 0 & [0.9,1.1] \\ 0 & [0.9,1.1] & [1.9,2.1] \\ [0.9,1.1] & [1.9,2.1] & [2.9,3.1] \end{pmatrix} \qquad x = \begin{pmatrix} [0.8,1.2] \\ [0.8,1.2] \\ [0.8,1.2] \end{pmatrix} \tag{7}$$

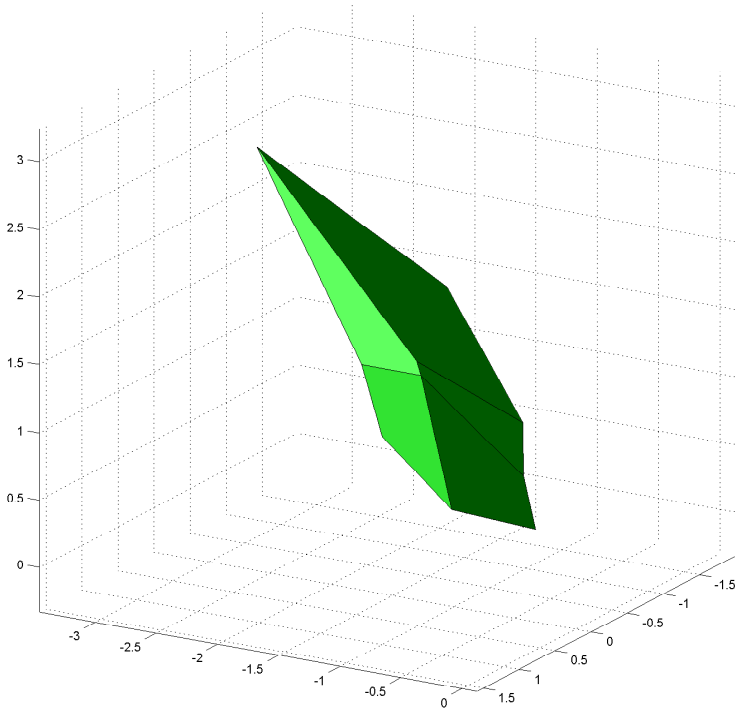

Figure 3: Solution set of the interval system (7).

4 Results of Numerical Experiments

4.1 Influence of the Basic Enclosure Methods

We have carried out a series of numerical experiments with the Neumaier interval linear systems (Example 1) changing the dimension n, the parameter θ, and the basic enclosure method *Encl*. In Table 1, we present the experimental results, namely the CPU run time in seconds (rounded to three digits after the decimal comma) spent for the estimation of the first component of the solution set.

All numerical tests have been carried out on a laptop with CPU Intel Core 2 Duo at 1.6 GGz, 800 MGz bus, and 1 Gb RAM (that is, with relatively modest capabilities).

The basic enclosure methods are denoted in the following way: `K` means the Krawczyk method, `MK` is the modified Krawczyk method with epsilon inflation, `G` is the interval Gauss method, `GS` is the interval Gauss-Seidel method, `HBR` is the Hansen-Bliek-Rohn procedure, and `V` is the procedure `verifylss` from INTLAB.

Our experimental results show that the Krawczyk method is the least effective among the basic enclosure methods. The programs based on the modified Krawczyk method, interval Gauss, and Gauss-Seidel methods work more quickly. However, the use of these methods as the basic enclosure methods cannot be recommended in the general case. Hansen-Bliek-Rohn procedure is the most effective as a basic enclosure method. The results of the procedure `verifylss` from INTLAB/MATLAB are worse, because the enclosure of the solution set is typically not found on the first stage; after

Table 1: Comparison of the basic enclosure methods (Example 1).

Parameter θ	Run time of the programs with different basic enclosure methods					
	K	MK	G	GS	HBR	V
$n = 5$						
10	3.314	2.847	2.814	2.818	1.046	2.009
14	0.386	0.326	0.700	0.714	0.284	0.338
22	0.210	0.229	0.500	0.571	0.195	0.238
26	0.212	0.199	0.499	0.539	0.189	0.207
30	0.214	0.197	0.499	0.547	0.188	0.204
$n = 8$						
16	390.255	103.098	303.903	289.908	65.099	89.632
24	3.811	2.604	6.305	6.544	2.004	1.881
32	1.384	1.357	3.479	3.578	1.104	1.038
40	0.894	0.962	2.265	2.283	0.688	0.649
48	0.743	0.763	1.948	1.971	0.589	0.557
$n = 10$						
22	3282.707	610.170	1142.017	1028.602	166.874	520.371
26	100.611	30.148	84.395	82.148	25.425	26.211
34	11.968	6.359	18.369	18.371	5.392	4.956
50	2.836	2.595	8.459	8.408	2.474	2.290
58	2.286	2.311	5.813	5.750	1.704	1.573
$n = 20$						
100	488.995	512.055	1104.751	1280.220	260.016	487.732
150	60.295	83.299	183.915	190.815	42.264	104.440
250	13.138	26.014	56.132	59.772	12.938	33.273
$n = 50$						
2000	261.104	1940.441	1104.713	1205.662	125.200	2155.334
7000	95.613	1240.579	325.509	350.331	64.227	1577.862

seven iterations of the Krawczyk method the Hansen-Bliek-Rohn procedure starts, so it is preferable to use the Hansen-Bliek-Rohn procedure as the basic enclosure method.

Changing the value of the parameter θ, one can analyze dependence of the run time on the properties of the interval matrix, i.e., its nearness to the singularity boundary. In Table 1, we present the run times of our numerical tests with the Neumaier interval linear system (Example 1) for various parameter values θ (we present only those values of θ for which the run time of the program is not too large). For near-zero ρ and sufficiently large $\Delta\sigma$, the run time of the program is small, but it increases exponentially for the matrices near the boundary of singularity, i.e., if $\rho \to 1$ and $\Delta\sigma \to 0$.

We characterized the properties of the interval matrix of the test system by the following quantities: the spectral radius ρ of the matrix $|(\operatorname{mid} \boldsymbol{A})^{-1}| \cdot \operatorname{rad} \boldsymbol{A}$ and the difference $\Delta\sigma$ between the least and the largest singular values of the matrices $\operatorname{mid} \boldsymbol{A}$ and $\operatorname{rad} \boldsymbol{A}$, respectively. The choice of these characteristics is motivated by the following well-known results.

Ris-Beeck criterion (see [17]). *If, for an interval $n \times n$-matrix $\boldsymbol{A}$, the midpoint matrix* $\operatorname{mid} \boldsymbol{A}$ *is regular and* $\rho\big(|(\operatorname{mid} \boldsymbol{A})^{-1}| \cdot \operatorname{rad} \boldsymbol{A}\big) < 1$, *then $\boldsymbol{A}$ is also regular.*

Rump criterion (see [17]). *If an interval $n \times n$-matrix $\boldsymbol{A}$ satisfies the inequality* $\sigma_{\max}(\operatorname{rad} \boldsymbol{A}) < \sigma_{\min}(\operatorname{mid} \boldsymbol{A})$, *then $\boldsymbol{A}$ is regular.*

Table 2: Characteristics ρ and $\Delta\sigma$ of the Neumaier system.

Parameter θ	Characteristics ρ and $\Delta\sigma$	
	$\rho\left(\lvert(\mathrm{mid}\ \boldsymbol{A})^{-1}\rvert \cdot \mathrm{rad}\ \boldsymbol{A}\right)$	$\sigma_{\min}(\mathrm{mid}\ \boldsymbol{A}) - \sigma_{\max}(\mathrm{rad}\ \boldsymbol{A})$
$n = 5$		
10	0.5397	5
14	0.3590	9
18	0.2674	13
22	0.2125	17
26	0.1760	21
30	0.1501	25
$n = 8$		
16	0.5884	8
20	0.4503	12
24	0.3633	16
28	0.3037	20
32	0.2605	24
36	0.2279	28
40	0.2024	32
44	0.1819	36
48	0.1652	40
$n = 10$		
24	0.5392	14
28	0.4423	18
32	0.3740	22
36	0.3235	26
40	0.2846	30
44	0.2539	34
48	0.2291	38
52	0.2086	42
56	0.1914	46
60	0.1767	50

In Table 2, the values of the characteristics ρ and $\Delta\sigma$ for the Neumaier interval linear system are displayed for the various values of its diagonal parameter θ and the system dimension n. We present these values to the fourth decimal digit.

Figures 4 and 5 show the run time dependencies on the characteristics ρ and $\Delta\sigma$ for the Neumaier 5×5-system with various basic enclosure methods.

In Table 3, we present the values of the characteristics ρ and $\Delta\sigma$ for the Shary interval linear system (Example 2) for various values of its parameters α, β, N, and the dimension n. For our test interval systems, we have carried out a number of numerical experiments with various basic enclosure methods. Table 3 displays the run time of the estimation of the first component of the solution sets.

Having analysed the experimental results, we draw conclusions that are quite similar to those for Example 1. The Krawczyk method and its modification are the least efficient as the basic enclosure method (we do not even present the corresponding results in Table 3). The program which uses the Hansen-Bliek-Rohn procedure has the

Table 3: Comparison of the basic enclosure methods (Example 2).

Parameter N of the interval matrix	Characteristic of the matrix		Run time of the programs with different basic enclosure methods			
	ρ	$\Delta\sigma$	`G`	`GS`	`HBR`	`V`
$\alpha = 0.4,\ \beta = 0.6,\ n = 10$						
15	0.6757	3.6	11.225	9.054	2.356	3.578
20	0.7353	3.6	10.823	9.322	2.341	3.853
25	0.7764	3.6	10.811	9.211	2.322	3.855
$\alpha = 0.4,\ \beta = 0.6,\ n = 20$						
25	0.6219	7.6	105.509	93.826	21.935	39.846
30	0.6637	7.6	105.558	97.270	21.959	25.755
35	0.6972	7.6	105.554	95.336	21.880	77.288
$\alpha = 0.4,\ \beta = 0.6,\ n = 30$						
35	0.6014	11.6	386.129	407.671	91.435	456.992
40	0.6329	11.6	386.522	426.977	91.432	358.437
45	0.6598	11.6	385.823	430.413	92.363	676.331
$\alpha = 0.6,\ \beta = 0.8,\ n = 10$						
15	0.5150	5.4	3.659	3.582	1.081	2.313
20	0.6029	5.4	7.915	6.450	1.622	1.876
25	0.6646	5.4	7.904	6.377	1.626	1.919
$\alpha = 0.6,\ \beta = 0.8,\ n = 20$						
25	0.4328	11.4	23.631	21.467	5.309	17.544
30	0.4956	11.4	23.892	21.540	5.310	47.364
35	0.5458	11.4	27.784	24.573	6.299	97.143
$\alpha = 0.6,\ \beta = 0.8,\ n = 30$						
35	0.4021	17.4	74.710	68.064	16.188	216.368
40	0.4494	17.4	74.648	67.998	16.192	441.926
45	0.4897	17.4	74.671	67.643	16.188	1616.238
$\alpha = 0.3,\ \beta = 0.7,\ n = 10$						
15	0.7353	2.7	13.542	12.483	3.373	5.410
20	0.7874	2.7	13.435	12.864	3.259	4.170
25	0.8224	2.7	13.457	12.525	3.360	4.186
$\alpha = 0.3,\ \beta = 0.7,\ n = 20$						
25	0.6868	5.7	157.602	151.104	34.394	34.016
30	0.7246	5.7	159.019	144.843	34.416	34.065
35	0.7543	5.7	161.887	141.492	34.346	37.167
$\alpha = 0.3,\ \beta = 0.7,\ n = 30$						
35	0.6679	8.7	712.694	656.571	149.929	145.106
40	0.6969	8.7	704.724	670.453	150.615	146.732
45	0.7212	8.7	712.297	647.965	151.154	147.519

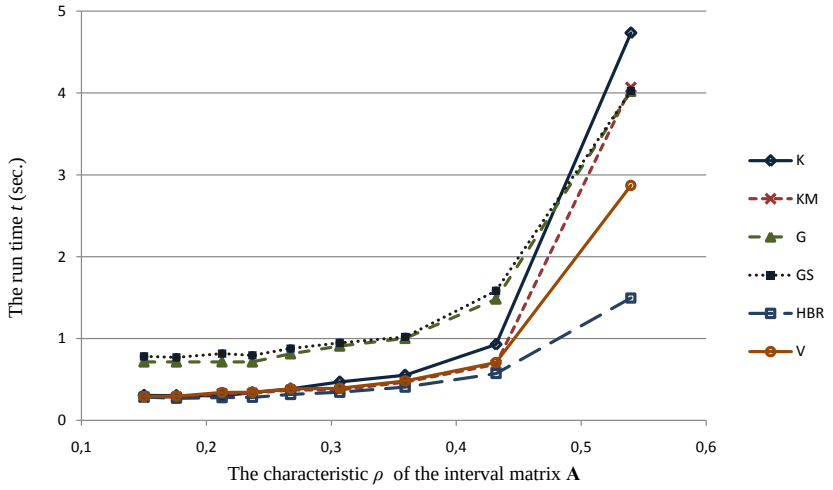

Figure 4: Run time dependence on the characteristic ρ for the Neumaier system.

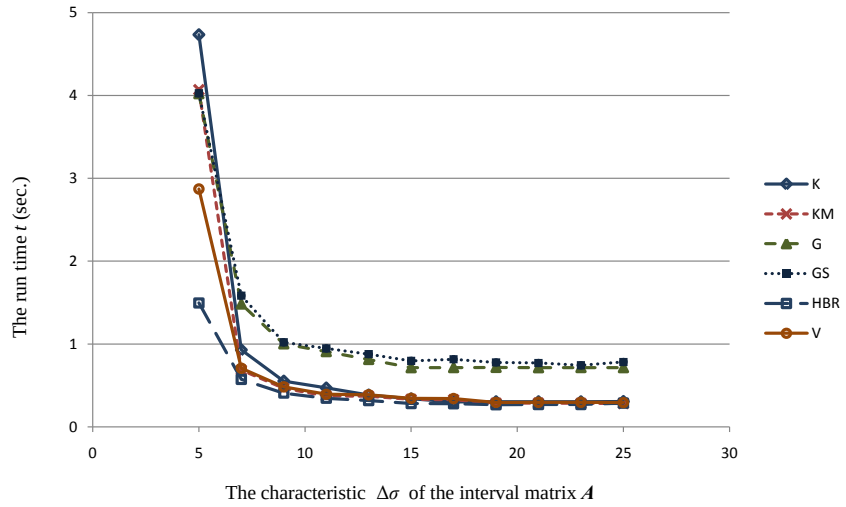

Figure 5: Run time dependence on the characteristic $\Delta\sigma$ for the Neumaier system.

best efficiency.

Notice that the efficiency of the program based on the procedure `verifylss` from INTLAB/MATLAB decreases significantly for systems of large dimension because of growing the run time of the Krawczyk iterations performed at its first stage. The programs using the interval Gauss and Gauss-Seidel methods demonstrated quite similar results. Table 3 shows that, for fixed values of the dimension n and parameters α and

Table 4: Comparison of the basic enclosure methods (Example 3).

Parameters $r = R$	Characteristic ρ	Run time of the programs with different basic enclosure methods			
		`G`	`GS`	`HBR`	`V`
$n = 5$					
0.1	0.2281	0.480	0.311	0.221	0.378
0.2	0.4562	0.422	0.375	0.116	0.199
0.3	0.6844	1.161	1.151	0.307	0.688
$n = 10$					
0.1	0.2028	0.813	0.810	0.200	0.216
0.2	0.4056	2.185	1.921	0.475	0.435
0.3	0.6083	4.083	5.345	0.820	1.399
$n = 15$					
0.1	0.1998	1.771	1.769	0.419	0.410
0.2	0.3997	5.861	5.101	1.154	0.838
0.3	0.5995	13.834	16.335	2.594	4.690
$n = 20$					
0.1	0.1991	3.107	3.094	0.696	0.678
0.2	0.3983	9.126	7.789	1.571	1.728
0.3	0.5974	41.162	43.119	7.576	11.403
$n = 25$					
0.1	0.199	4.863	4.805	1.062	1.592
0.2	0.3979	14.129	12.084	1.694	2.016
0.3	0.5969	90.467	104.553	16.273	27.129
$n = 30$					
0.1	0.1989	10.969	10.546	1.557	2.238
0.2	0.3979	22.807	19.828	3.450	4.110
0.3	0.5968	216.787	231.572	39.057	60.433

β, increasing the parameter N does not result in change of the characteristic $\Delta\sigma$ of the Shary test system, while the characteristic ρ varies insignificantly. Hence, the run times of the programs based on the interval Gauss method, the interval Gauss-Seidel method, and the Hansen-Bliek-Rohn procedure differ insignificantly, but the program modification using the procedure `verifylss` is sensitive to the change of the parameter N of the interval matrix $\boldsymbol{A}$ (see Table 3) because the interval enclosure of the solution set is found usually on the first stage of `verifylss`, i.e., by the modified Krawczyk method, and the accuracy of the enclosure significantly depends on the parameter N.

Next, we carried out numerical tests with the interval system from Example 3 and examined the efficiency of the algorithms based on various basic enclosure methods (see Table 4 and Figure 6). Having analysed the experimental results, we can make conclusions that are analogous to the previous ones.

We have investigated various versions of PPS-methods, using basic enclosure methods: the Krawczyk method, the modified Krawczyk method with epsilon inflation, the interval Gauss method, the interval Gauss-Seidel iteration, the Hansen-Bliek-Rohn procedure, and the `verifylss` procedure from the toolbox INTLAB. Experimental results demonstrated that, amongst these techniques, the Hansen-Bliek-Rohn procedure

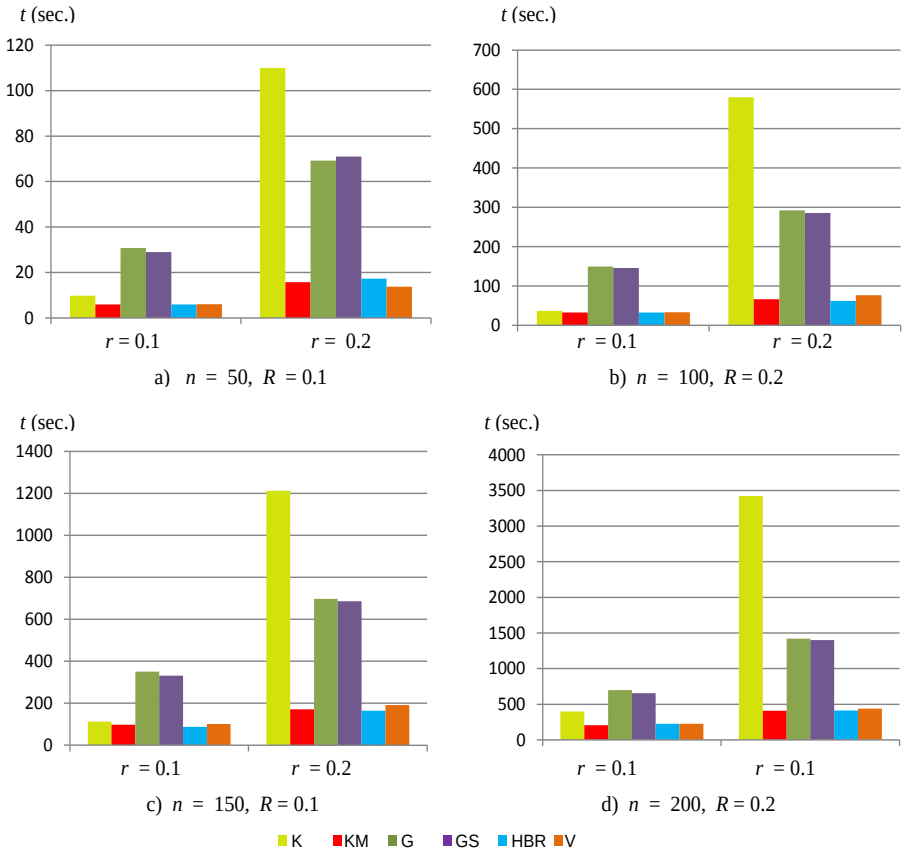

Figure 6: The run time of the estimation for the first component of the solution set to the Toft interval linear system (Example 3) for various values of n, r, and R.

with preliminary preconditioning is the best basic enclosure method for PPS-methods. The run times of the program depend on the properties of the interval matrix of the system, i.e., the nearness to the boundary of singularity and the dimension of the interval matrix. Recall that we characterized the properties of the interval matrix of the system by the quantities $\rho = \rho\left(|(\mathrm{mid}\ \boldsymbol{A})^{-1}| \cdot \mathrm{rad}\ \boldsymbol{A}\right)$ and $\Delta\sigma = \sigma_{\min}(\mathrm{mid}\ \boldsymbol{A}) - \sigma_{\max}(\mathrm{rad}\ \boldsymbol{A})$.

4.2 Influence of Structure of the Working lList and its Processing

In Section 2.3, we described four modes of processing of the working list $\mathcal{L}$, in which the results of the partitioning of the interval linear system are stored:

1) the list $\mathcal{L}$ is an unordered list of records (a heap);
2) the records of the list $\mathcal{L}$ are in ascending ordered by the estimate $\Upsilon(\boldsymbol{Q}, \boldsymbol{r})$;

3) in the list $\mathcal{L}$, an ordered sublist $\mathcal{L}_l$ of the active records is separated, having a fixed maximal length, and the remaining records are stored as a heap;
4) Pankov's mode [15].

We have experimentally investigated the modifications of the algorithm for optimal outer estimation of the solution set to the interval linear system, which implemented the above modes of the working list processing, and tested them on the interval systems from Examples 1–3. We used the Hansen-Bliek-Rohn procedure as the basic enclosure method. The run times (seconds) for estimating the first component of the solution set are presented in Table 5.

Table 5: The efficiency of the algorithms for processing the working list.

Parameters and dimension of the interval system		Run time of the programs			
		1	2	3	4
Neumaier interval linear system from Example 1					
$n = 6$	$\theta = 12$	4.347	3.013	2.082	2.943
	$\theta = 16$	1.000	0.677	0.667	0.670
	$\theta = 20$	0.596	0.404	0.399	0.400
$n = 8$	$\theta = 16$	119.767	73.602	65.039	72.844
	$\theta = 20$	10.241	5.452	5.432	5.427
	$\theta = 24$	3.859	2.036	2.004	2.031
$n = 10$	$\theta = 20$	5815.058	2323.042	1407.043	2235.154
	$\theta = 22$	336.505	185.925	166.874	184.065
	$\theta = 24$	67.605	41.349	40.886	41.154
$n = 12$	$\theta = 28$	853.403	461.624	420.766	460.146
	$\theta = 30$	301.189	178.815	175.215	176.626
	$\theta = 35$	84.595	48.643	48.003	48.689
Shary interval linear system from Example 2					
$n = 6$	$N = 20, n = 10$	5.739	3.428	3.259	3.275
	$N = 30, n = 20$	67.014	35.038	34.416	35.523
	$N = 40, n = 30$	297.970	151.301	150.615	151.303
Toft interval linear system from Example 3					
$n = 8$	$n = 10$	1.340	0.856	0.820	0.857
	$n = 20$	13.358	7.772	7.576	7.784
	$n = 30$	71.625	39.921	39.057	39.930

The processing of the working list formed as a heap is the least effective (mode 1) because looking through the whole list to find the leading record with the minimal estimate $\Upsilon(\boldsymbol{Q}, \boldsymbol{r})$ takes some time. List processing speed considerably increases if the records are in ascending order with respect to the estimate $\Upsilon(\boldsymbol{Q}, \boldsymbol{r})$ (mode 2).

Sorting the list, adding, and deleting the records from it take some time, so ordering and cleaning up the entire list $\mathcal{L}$ may prove of no use. It is advisable to handle only a part of it, namely the sublist $\mathcal{L}_l$ or $\mathcal{L}_\gamma$ of the active records (modes 3 and 4). Comparing modes 3 and 4, one can notice that mode 3 is preferable because the maximal length of the active records sublist is fixed. In Pankov's mode [15], the length of the sublist of the active records is defined by a threshold constant γ. This constant may be sufficiently large on some steps of the algorithm, which leads to reduction of its efficiency especially in case the matrix of the interval system is near the singularity boundary.

Note that if the working list is arranged according to Pankov's mode, the leading record has the smallest estimate $\Upsilon(\boldsymbol{Q}, \boldsymbol{r})$. This property may be violated for the active records sublist with a fixed maximal length (mode 3) that results in increasing the number of the algorithm steps.

4.3 Comparison of the Codes for Optimal Outer Estimation of the Solution Set

We have developed a free MATLAB/INTLAB code `linppsr` [25] implementing the PPS-method with Rohn's modification in which

- the Hansen-Bliek-Rohn procedure is used as the basic enclosure method,
- in the working list, an ordered sublist of active records having a fixed maximal length is separated, while the remaining records are unordered, forming a heap.

We compare `linppsr` with the algorithm `verintervalhull`, a procedure from the toolbox VERSOFT [26] based on Rohn's method. The results of our numerical experiments for Examples 1–3 with various dimensions and parameters are presented in Table 6. Figure 7 presents the results of numerical experiments with the Neumaier system (Example 1) and the Toft interval system (Example 3).

Table 6: The efficiency of the algorithms `verintervalhull` and `linppsr`.

Parameters and dimension of the interval system		Run time of the algorithms	
		`linppsr`	`verintervalhull`
Neumaier interval linear system from Example 1			
$n = 5$	$\theta = 10$	8.012	3.017
	$\theta = 20$	1.642	3.019
	$\theta = 30$	1.641	3.025
$n = 10$	$\theta = 25$	617.238	144.113
	$\theta = 30$	197.024	144.226
	$\theta = 45$	51.422	144.815
	$\theta = 60$	31.065	145.012
$n = 12$	$\theta = 30$	2943.612	1084.556
	$\theta = 50$	246.804	1084.476
	$\theta = 70$	99.308	1085.555
	$\theta = 90$	57.049	1084.001
Shary interval linear system from Example 2			
$\alpha = 0.4, \beta = 0.6$	$n = 10, N = 15$	42.528	144.717
	$n = 20, N = 25$	882.344	more than 8 hours
	$n = 30, N = 35$	5483.106	more than 8 hours
$\alpha = 0.6, \beta = 0.8$	$n = 10, N = 15$	17.477	142.084
	$n = 20, N = 25$	209.250	more than 8 hours
	$n = 30, N = 35$	969.280	more than 8 hours
Toft interval linear system from Example 3			
$r = 0.2$	$n = 10$	5.750	9.115
	$n = 20$	56.524	1321.249
	$n = 30$	161.393	more than 8 hours

Having analysed the experimental results, we can draw the following conclusions:

- The run time of both algorithms grows exponentially with the system dimension.
- The run time of the procedure `verintervalhull` does not depend on the properties of the interval matrix $\boldsymbol{A}$ and nearly constant for different interval systems of the same size.
- The speed of the algorithm `linppsr` slows down as the matrix $\boldsymbol{A}$ approaches the boundary of singularity.
- For large dimensions n, the procedure `linppsr` is definitely more efficient than `verintervalhull`.

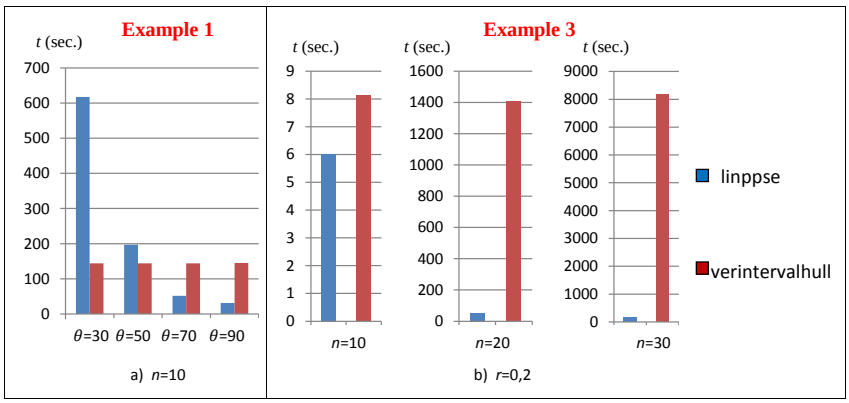

Figure 7: Comparison of the algorithms `linppsr` and `verintervalhull` for the optimal outer estimation of the solution set.

Since the desired estimates $\min\{\, x_\nu \mid x \in \Xi(\boldsymbol{A},\boldsymbol{b})\}$ and $\max\{\, x_\nu \mid x \in \Xi(\boldsymbol{A},\boldsymbol{b})\}$, $\nu = 1, 2, \ldots, n$, are reached at the set of at most 2^n extreme solutions, the upper bound of the computational complexity of Rohn's method is 2^n. The computational complexity of the PPS-methods without the Rohn modification is proportional to 2^{n^2} at worst, but PPS-methods are less laborious on the average, because they flexibly adjust their execution to the problem under solution, while their modifications (monotonicity test, etc.) prevent them from realizing the worst computational scenario. Every step of the PPS-methods essentially uses information obtained from the previous steps, so these methods are "execution-driven". In formal terms, one can say that PPS-methods have *adaptive* computational schemes.

Yet another remarkable feature of PPS-methods is that they provide a *sequential guarantee* of the results produced during their execution [3, 21]. This means that PPS-methods generate a sequence of approximate estimates for $\min\{\, x_\nu \mid x \in \Xi(\boldsymbol{A},\boldsymbol{b})\}$ and $\max\{\, x_\nu \mid x \in \Xi(\boldsymbol{A},\boldsymbol{b})\}$, $\nu = 1, 2, \ldots, n$, from below and from above, respectively, that is, from the "necessary sides" prescribed by the problem statement. Then, if the dimension of the system is large or its matrix is near the boundary of singularity, and the problem under solution is laborious and there are not enough computational resources for its completion, we can stop the algorithm before its natural completion

and report a correct answer to the problem. We get estimates which may be not optimal, but they provide us with (more or less exact) *outer* approximations of the required $\min\{ x_\nu \mid x \in \Xi(\boldsymbol{A}, \boldsymbol{b})\}$ and $\max\{ x_\nu \mid x \in \Xi(\boldsymbol{A}, \boldsymbol{b})\}$, $\nu = 1, 2, \ldots, n$. Artificial intelligence researchers call such algorithms *anytime algorithms* (see e.g. [1]).

The program `verintervalhull` provides merely a *final guarantee* of the result it produces. That is, `verintervalhull` computes a correct enclosure of the solution set only after its natural completion, which may require enormous time for large problems. As a consequence, `verintervalhull` and the algorithm it implements are impractical for the solution of interval linear systems with the dimension greater than 20–30.

5 Acknowledgements

The authors are grateful to Irene Sharaya for the development of the drawing package IntLinInc3D [27] and for her assistance in plotting the solution sets in this paper.

References

[1] Anytime algorithm, Wikipedia article, available at `http://en.wikipedia.org/wiki/Anytime_algorithm`.

[2] H. Beeck. Über die Struktur und Abschätzungen der Lösungsmenge von linearen Gleichungssystemen mit Intervallkoeffizienten, *Computing*, vol. 10 (1972), pp. 231–244.

[3] M. Beltran, G. Castillo, and V. Kreinovich. Algorithms that still produce a solution (maybe not optimal) when interrupted: Shary's idea justified, *Reliable Computing*, vol. 4 (1998), pp. 39–53.

[4] M. Fiedler, J. Nedoma, J. Ramik, J. Rohn, and K. Zimmermann. *Linear Optimization Problems with Inexact Data.* New York, Springer, 2006.

[5] R.T. Gregory and D.L. Karney. *A Collection of Matrices for Testing Computational Algorithms.* Huntington, NY, Robert E. Krieger Publishing, 1978.

[6] E.R. Hansen. On linear algebraic equations with interval coefficients. In E. Hansen, editor, *Topics in Interval Analysis*, Oxford, Clarendon Press, 1969, pp. 35–46. Available at `http://www.nsc.ru/interval/Library/ProcBooks/IntvalTopics69.pdf`.

[7] G.I. Hargreaves. *Interval Analysis in MATLAB*, Manchester Center for Computational Mathematics, 2002. Available at `http://eprints.ma.man.ac.uk/1204`.

[8] R. Krawczyk. Newton-Algorithmen zur Besstimmung von Nullstellen mit Fehlerschranken, *Computing*, vol. 4 (1969), pp. 187–201.

[9] V. Kreinovich, A.V. Lakeyev, J. Rohn, and P. Kahl. *Computational Complexity and Feasibility of Data Processing and Interval Computations.* Dordrecht: Kluwer, 1997.

[10] A. Neumaier. *Interval Methods for Systems of Equations.* Cambridge, Cambridge University Press, 1990.

[11] A. Neumaier. A simple derivation of Hansen–Bliek–Rohn–Ning–Kearfott enclosure for linear interval equations, *Reliable Computing*, vol. 5 (1999), pp. 131–136.

[12] K. Nickel. Die Überschatzung des Wertebereiches einer Funktion in der Intervallrechnung mit Anwendungen auf lineare Gleichungssysteme, *Computing*, vol. 18 (1977), pp. 15–36.

[13] W. Oettli. On the solution set of linear system with inaccurate coefficients, *SIAM J. Numer. Analysis*, vol. 2 (1965), pp. 115–118.

[14] W. Oettli and W. Prager. Compatibility of approximate solution of linear equations with given error bounds for coefficients and right-hand sides, *Numerische Mathematik*, vol. 6 (1964), pp. 405–409.

[15] P.S. Pankov. Algorithms for verification of stable states and global optimization in bounded region. Frunze, 1984 (Deposited in VINITI, No. 5250-84Dep). (in Russian)

[16] J. Rohn. Systems of linear interval equations, *Linear Algebra and its Applications*, vol. 126 (1989), pp. 39–78.

[17] J. Rohn. *A Handbook of Results on Interval Linear Problems.* Technical Report No. 1163, Institute of Computer Science, Academy of Sciences of the Czech Republic, Prague, 2012. Available at `http://uivtx.cs.cas.cz/~rohn/publist/!aahandbook.pdf`.

[18] S.M. Rump. INTLAB – INTerval LABoratory, In T. Csendes, editor, *Developments in Reliable Computing*, Kluwer, Dordrecht, 1999, pp. 77–104.

[19] S.M. Rump. Verified solution of large systems and global optimization problems, *J. of Comput. and Applied Math.*, vol. 60 (1995), pp. 201–218.

[20] S.P. Shary. A new class of algorithms for optimal solution of interval linear systems, *Interval Computations*, No. 2(4) (1992), pp. 18–29.

[21] S.P. Shary. On optimal solution of interval linear equations. *SIAM J. Numer. Analysis*, vol. 32 (1995), pp. 610–630.

[22] S.P. Shary. *Finite-dimensional Interval Analysis.* Novosibirsk, XYZ, 2013. Electronic book available at `http://www-sbras.nsc.ru/interval/Library/InteBooks/SharyBook.pdf`. (in Russian)

[23] S.P. Shary. Parameter partition methods for optimal numerical solution of interval linear systems, In E. Krause, Yu.I. Shokin, M. Resch, N.Yu. Shokina, editors, *Computational Science and High-Performance Computing III. The 3rd Russian-German Advanced Research Workshop*, Novosibirsk, Russia, 23-27 July, 2007, Berlin-Heidelberg, Springer, 2008, pp. 184–205. Available at `http://www.nsc.ru/interval/shary/Papers/PPSmethods.pdf`.

[24] O. Toft. *Sequential and Parallel Solution of Linear Interval Equations*, Eksamensproject: NI-E-92-04, Numerisk Institute, Danmarks Tekniske Højskole. Lyngby, 1992.

[25] `linppsr` — a program implementing PPS-method for optimal (exact) outer estimation of the solution sets to interval linear systems. Available at `http://www.nsc.ru/interval/Programing/MCodes/linppsr.m`.

[26] J. Rohn. VERSOFT: Verification software in MATLAB/INTLAB. Available at `http://www.cs.cas.cz/rohn/matlab`.

[27] I.A. Sharaya. IntLinInc3D, a software package for visualization of solution sets to interval and noninterval 3D systems of linear relations. Available from `http://interval.ict.nsc.ru/sharaya`.