NOTES ON NUMERICAL FLUID MECHANICS AND MULTIDISCIPLINARY DESIGN · VOLUME 101

Computational Science and High Performance Computing III

The 3rd Russian-German Advanced Research Workshop, Novosibirsk, Russia, 23–27 July 2007

Egon Krause • Yurii I. Shokin
Michael Resch • Nina Shokina (Eds.)

Springer

Prof. Egon Krause
Aerodynamisches Institut
RWTH Aachen
Wuellnerstr. zw. 5 u. 7
52062 Aachen
Germany

Prof. Yurii I. Shokin
Academician of Russian Academy of Sciences
Institute of Computational Technologies of SB RAS
Ac. Lavrentyev Ave. 6
630090 Novosibirsk
Russia

Prof. Michael Resch
High Performance Computing Center Stuttgart
University of Stuttgart
Nobelstrasse 19
70569 Stuttgart
Germany

Dr. Nina Shokina
High Performance Computing Center Stuttgart
University of Stuttgart
Nobelstrasse 19
70569 Stuttgart
Germany

ISBN 978-3-540-69008-5 e-ISBN 978-3-540-69010-8

DOI 10.1007/978-3-540-69010-8

Notes on Numerical Fluid Mechanics and Multidisciplinary Design ISSN 1612-2909

Library of Congress Control Number: 2008928447

Typeset & Cover Design: Scientific Publishing Services Pvt. Ltd., Chennai, India.

Printed in acid-free paper

5 4 3 2 1 0

springer.com

Parameter Partition Methods for Optimal Numerical Solution of Interval Linear Systems

S.P. Shary

Institute of computational technologies SB RAS,
Lavrentiev ave. 6, 630090 Novosibirsk, Russia
shary@ict.nsc.ru

Abstract. The paper presents a new class of adaptive and sequentially guaranteeing *PPS-methods*, based on *partitioning parameter sets*, for computing optimal (exact) component-wise bounds of the solution sets to interval linear systems with square regular matrices.

1 Introduction

The subject of the present work is the problem of outer interval estimation of the solution set to an interval linear system

$$\boldsymbol{A}x = \boldsymbol{b} \tag{1}$$

with a regular (nonsingular) interval $n \times n$-matrix $\boldsymbol{A} = (\,\boldsymbol{a}_{ij})$ and an interval right-hand side n-vector $\boldsymbol{b} = (\,\boldsymbol{b}_i)$. The *solution set* of the interval linear system (1) is known to be defined the set

$$\Xi(\boldsymbol{A},\boldsymbol{b}) := \big\{\, x \in \mathbb{R}^n \mid (\exists A \in \boldsymbol{A})(\exists b \in \boldsymbol{b})(\,Ax = b\,) \,\big\}, \tag{2}$$

formed by solutions to all the point systems $Ax = b$ with $A \in \boldsymbol{A}$ and $b \in \boldsymbol{b}$. $\Xi(\boldsymbol{A},\boldsymbol{b})$ is often referred to as *united solution set* insofar as there exist a variety of other solution sets to interval systems of equations (see e.g. [1]). We will not consider them in our paper, so that the united solution set will be the only one being studied. In what follows, we shall thereby call it just solution set.

An exact description of the solution set is practically impossible for the dimensions n larger than several tens, since its complexity grows exponentially with n. On the other hand, such an exact description is not really necessary in most cases. The users traditionally confine themselves to computing some *estimates*, in a prescribed sense, of the solution set, and below we are going to solve the following problem of *outer* (by supersets) interval estimation:

$$\boxed{\begin{array}{l}\textit{For an interval system of linear equations } \boldsymbol{A}x = \boldsymbol{b},\\ \textit{find an interval enclosure of the solution set } \Xi(\boldsymbol{A},\boldsymbol{b}).\end{array}} \tag{3}$$

E. Krause et al. (Eds.): Comp. Science & High Perf. Computing III, NNFM 101, pp. 184–205, 2008.
springerlink.com

Sometimes, a component-wise form of the problem (3) is considered:

$$\boxed{\begin{array}{l}\textit{For an interval system of linear equations } \boldsymbol{A}x = \boldsymbol{b},\\ \textit{find estimates for } \min\{\, x_\nu \mid x \in \Xi(\boldsymbol{A},\boldsymbol{b}) \,\} \textit{ from below,}\\ \textit{for } \max\{\, x_\nu \mid x \in \Xi(\boldsymbol{A},\boldsymbol{b}) \,\} \textit{ from above, } \nu = 1,2,\ldots,n.\end{array}} \tag{4}$$

One can find the extensive (but not at all exhaustive) bibliography on the problem (3)–(4) e.g. in [2,3,4,5,6,7].

Practical needs often require that the solution to the interval problem should be not any one, but *optimal*, i.e. the best in some sense. It is fairly simple to realize that the optimal solution to (3)–(4) is the *interval hull* of the solution set, that is, the least inclusive interval vector guaranteed to contain the solution set. The optimality requirement makes the problem statement (3)–(4) NP-hard in general, if we do not restrict the widths of the intervals in the system and/or the structure of nonzero elements in the matrix $\boldsymbol{A}$ [8]. Still, in the present work we advance an efficient *adaptive* numerical technique — parameter partitioning methods or PPS-method — for computing such optimal outer estimates of the solution sets for interval linear systems.

Our notation follows, in the major lines, an informal international standard [9]. In particular, we designate intervals and interval objects by boldface letters, while underbars and overbars mean lower and upper endpoints of the corresponding intervals. The set of all intervals is denoted by $\mathbb{IR}$, and we identify real numbers with zero-width degenerate intervals.

2 Parameter Partition Method for Interval Linear Systems

In the rest of the paper, we concentrate on computing $\min\{\, x_\nu \mid x \in \Xi(\boldsymbol{A},\boldsymbol{b}) \,\}$ for a fixed integer index $\nu \in \{\, 1,2,\ldots,n \,\}$, since

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

Let

Encl be a method that computes an enclosure of the solution set (we shall call it *basic method*),

Encl$\,(\boldsymbol{Q},\boldsymbol{r})$ be an interval enclosure, produced by the method *Encl*, of the solution set $\Xi(\boldsymbol{Q},\boldsymbol{r})$ to the system $\boldsymbol{Q}x = \boldsymbol{r}$, that is, *Encl*$\,(\boldsymbol{Q},\boldsymbol{r}) \in \mathbb{IR}^n$ and

$$\textit{Encl}\,(\boldsymbol{Q},\boldsymbol{r}) \supseteq \Xi(\boldsymbol{Q},\boldsymbol{r}),$$

$\Upsilon(\boldsymbol{Q},\boldsymbol{r})$ be the lower endpoint of the ν-th component of the interval enclosure for the solution set $\Xi(\boldsymbol{Q},\boldsymbol{r})$ obtained by the method *Encl*, that is,

$$\Upsilon(\boldsymbol{Q},\boldsymbol{r}) := \underline{\big(\,\textit{Encl}\,(\boldsymbol{Q},\boldsymbol{r})\,\big)_\nu}. \tag{5}$$

We require that the basic method should satisfy

> the estimate $\Upsilon(\boldsymbol{Q},\boldsymbol{r})$ is inclusion monotonic with respect to the matrix $\boldsymbol{Q}$ and vector $\boldsymbol{r}$, i.e., for all $\boldsymbol{Q}'$, $\boldsymbol{Q}''\in\mathbb{IR}^{n\times n}$ and $\boldsymbol{r}'$, $\boldsymbol{r}''\in\mathbb{IR}^n$, $\boldsymbol{Q}'\subseteq\boldsymbol{Q}''$ and $\boldsymbol{r}'\subseteq\boldsymbol{r}''$ implies the inequality $\Upsilon(\boldsymbol{Q}'',\boldsymbol{r}'')\le\Upsilon(\boldsymbol{Q}',\boldsymbol{r}')$. (C1)

For most of the popular techniques computing enclosures of the solution set to interval linear systems (interval Gauss method [2,5], interval Gauss-Seidel iteration [4,5], various modifications of the stationary iterative single-step and total-step techniques [2], Krawczyk method [5], etc.), the fulfillment of (C1) can be easily derived from the inclusion monotonicity of the interval arithmetic operations.

To go further, we need to remind a remarkable result first obtained by H. Beeck [10] and afterward repeatedly proved by K. Nickel [11]: if $\boldsymbol{A}$ is regular (i.e., contains only regular point matrices), then both minimal and maximal componentwise values of the points from the solution set are attained at the so-called *extreme* matrices and right-hand side vectors made up of the *endpoints* of $\boldsymbol{A}$ and $\boldsymbol{b}$. In other words, for any $\nu=1,2,\ldots,n$,

$$\min\{\,x_\nu \mid x\in\Xi(\boldsymbol{A},\boldsymbol{b})\,\}=\big(\tilde{A}^{-1}\tilde{b}\,\big)_\nu$$

with a point matrix $\tilde{A}\in\mathbb{R}^{n\times n}$ and a point vector $\tilde{b}\in\mathbb{R}^n$ whose elements are the endpoints of the interval entries of the matrix $\boldsymbol{A}$ and vector $\boldsymbol{b}$ respectively. It is also worth noting that

$$\Upsilon(\tilde{A},\tilde{b})\le\big(\tilde{A}^{-1}\tilde{b}\,\big)_\nu$$

due to the very definition of the estimate Υ.

Assuming that an entry $\boldsymbol{a}_{ij}$ of the matrix $\boldsymbol{A}$ has nonzero width, we denote

> by $\boldsymbol{A}'$ and $\boldsymbol{A}''$ the matrices obtained from $\boldsymbol{A}$ through replacing the entry $\boldsymbol{a}_{ij}$ by $\underline{\boldsymbol{a}}_{ij}$ and $\overline{\boldsymbol{a}}_{ij}$ respectively, (6)
>
> by A' and A'' the matrices obtained from $\tilde{A}$ through replacing the entry $\tilde{a}_{ij}$ by $\underline{\boldsymbol{a}}_{ij}$ and $\overline{\boldsymbol{a}}_{ij}$ respectively.

Inasmuch as

$$A'\subseteq\boldsymbol{A}'\subseteq\boldsymbol{A},\qquad\qquad A''\subseteq\boldsymbol{A}''\subseteq\boldsymbol{A},$$

and $\tilde{b}\subseteq\boldsymbol{b}$, the condition (C1) implies the inequalities

$$\Upsilon(\boldsymbol{A},\boldsymbol{b})\le\Upsilon(\boldsymbol{A}',\boldsymbol{b})\le\Upsilon(A',\tilde{b})\quad\text{ and }\quad\Upsilon(\boldsymbol{A},\boldsymbol{b})\le\Upsilon(\boldsymbol{A}'',\boldsymbol{b})\le\Upsilon(A'',\tilde{b}).$$

Therefore, taking the minima of the corresponding inequality sides, we arrive at

$$\Upsilon(\boldsymbol{A},\boldsymbol{b})\;\le\;\min\Big\{\,\Upsilon(\boldsymbol{A}',\boldsymbol{b}),\,\Upsilon(\boldsymbol{A}'',\boldsymbol{b})\,\Big\}\;\le\;\min\Big\{\,\Upsilon(A',\tilde{b}),\,\Upsilon(A'',\tilde{b})\,\Big\}.$$

Additionally,

$$\min\Big\{ \Upsilon(A', \tilde{b}),\, \Upsilon(A'', \tilde{b}) \Big\} \;\le\; \Upsilon(\tilde{A}, \tilde{b}) \;\le\; \Big(\tilde{A}^{-1}\tilde{b} \Big)_{\nu} \;=\; \min\{\, x_\nu \mid x \in \Xi(\boldsymbol{A}, \boldsymbol{b}) \,\}.$$

Comparing the above two inequality chains results in the relation

$$\Upsilon(\boldsymbol{A}, \boldsymbol{b}) \;\le\; \min\Big\{ \Upsilon(\boldsymbol{A}', \boldsymbol{b}),\, \Upsilon(\boldsymbol{A}'', \boldsymbol{b}) \Big\} \;\le\; \min\{\, x_\nu \mid x \in \Xi(\boldsymbol{A}, \boldsymbol{b}) \,\},$$

and, as a consequence, in the following practical prescription: *having solved the two interval "systems-descendants"* $\boldsymbol{A}'x = \boldsymbol{b}$ *and* $\boldsymbol{A}''x = \boldsymbol{b}$ *defined by (6) we can get better estimate for* $\min\{\, x_\nu \mid x \in \Xi(\boldsymbol{A}, \boldsymbol{b}) \,\}$ *from below as* $\min\{\, \Upsilon(\boldsymbol{A}', \boldsymbol{b}),\, \Upsilon(\boldsymbol{A}'', \boldsymbol{b}) \,\}$.

In the right-hand side vector $\boldsymbol{b}$, breaking an interval element $\boldsymbol{b}_i$ up into its endpoints $\underline{\boldsymbol{b}}_i$ and $\overline{\boldsymbol{b}}_i$ has the similar effect. For uniformity, we will designate by $\boldsymbol{A}'x = \boldsymbol{b}'$ and $\boldsymbol{A}''x = \boldsymbol{b}''$ the interval "systems-descendants" we get from $\boldsymbol{A}x = \boldsymbol{b}$ after having subdivided an interval element of either the matrix $\boldsymbol{A}$ or right-hand side vector $\boldsymbol{b}$.

To further improve the estimate for $\min\{\, x_\nu \mid x \in \Xi(\boldsymbol{A}, \boldsymbol{b}) \,\}$, it makes sense to repeat the above described subdivision procedure applying it to the "systems-descendants" $\boldsymbol{A}'x = \boldsymbol{b}'$ and $\boldsymbol{A}''x = \boldsymbol{b}''$, and then to subdivide the descendants of $\boldsymbol{A}'x = \boldsymbol{b}'$ and $\boldsymbol{A}''x = \boldsymbol{b}''$ again to get even better estimate, and so forth. We arrange the whole process of the successive step-by-step improvement of the estimate for $\min\{\, x_\nu \mid x \in \Xi(\boldsymbol{A}, \boldsymbol{b}) \,\}$ in accordance with the well-known "branch-and-bound" technique, similar to that implemented in the popular interval global optimization methods from [3,4,12] and Lipschitz global optimization methods from [13]:

- first, all the interval systems $\boldsymbol{Q}x = \boldsymbol{r}$ emerging as the result of the partitioning of the original system (1) as well as their estimates $\Upsilon(\boldsymbol{Q}, \boldsymbol{r})$ are stored in a *working list* $\mathcal{L}$;
- second, at every step of our algorithm, the interval system subject to subdivision is that providing the smallest current estimate $\Upsilon(\boldsymbol{Q}, \boldsymbol{r})$;
- third, the interval element to be subdivided in the system $\boldsymbol{Q}x = \boldsymbol{r}$ is the one having the maximal width.

The execution of the algorithm thus amounts to maintaining the list $\mathcal{L}$, which consists of records having the form of *triples*

$$\Big(\boldsymbol{Q},\; \boldsymbol{r},\; \Upsilon(\boldsymbol{Q}, \boldsymbol{r}) \Big), \tag{7}$$

where $\boldsymbol{Q}$ is an interval $n \times n$-matrix, $\boldsymbol{Q} \subseteq \boldsymbol{A}$,
$\boldsymbol{r}$ is an interval n-vector, $\boldsymbol{r} \subseteq \boldsymbol{b}$.

Besides, the records forming the working list $\mathcal{L}$ will be ordered with respect to the values of the estimate $\Upsilon(\boldsymbol{Q}, \boldsymbol{r})$, while the first record of $\mathcal{L}$ as well as the corresponding interval system $\boldsymbol{Q}x = \boldsymbol{r}$ and the estimate Υ (the smallest in the list) will be called *leading* ones at the current step of the method. Table 1

Table 1. The simplest PPS-method for interval linear systems

Input

An interval linear system $\boldsymbol{A}x = \boldsymbol{b}$.

A number ν of the component estimated.

A method *Encl* that produces the estimate Υ by the rule (5).

Output

An estimate Z for $\min\{\, x_\nu \mid x \in \Xi(\boldsymbol{A}, \boldsymbol{b}) \,\}$ from below.

Algorithm

assign $\boldsymbol{Q} := \boldsymbol{A}$ and $\boldsymbol{r} := \boldsymbol{b}$;

compute the estimate $\upsilon := \Upsilon(\boldsymbol{Q}, \boldsymbol{r})$;

initialize the list $\mathcal{L} := \{\, (\boldsymbol{Q}, \boldsymbol{r}, \upsilon) \,\}$;

`DO WHILE` (either $\boldsymbol{Q}$ or $\boldsymbol{r}$ has an interval entry)

in the matrix $\boldsymbol{Q} = (\,\boldsymbol{q}_{ij})$ and vector $\boldsymbol{r} = (\,\boldsymbol{r}_i)$, choose an interval element $\boldsymbol{h}$ having the maximal width;

generate interval systems $\boldsymbol{Q}'x = \boldsymbol{r}'$ and $\boldsymbol{Q}''x = \boldsymbol{r}''$ so that

if $\boldsymbol{h} = \boldsymbol{q}_{kl}$ for some $k, l \in \{\, 1, 2, \ldots, n \,\}$, then set

$$\boldsymbol{q}'_{ij} := \boldsymbol{q}''_{ij} := \boldsymbol{q}_{ij} \text{ for } (i,j) \neq (k,l),$$
$$\boldsymbol{q}'_{kl} := \underline{\boldsymbol{q}}_{kl}, \quad \boldsymbol{q}''_{kl} := \overline{\boldsymbol{q}}_{kl}, \quad \boldsymbol{r}' := \boldsymbol{r}'' := \boldsymbol{r};$$

if $\boldsymbol{h} = \boldsymbol{r}_k$ for some $k \in \{\, 1, 2, \ldots, n \,\}$, then set

$$\boldsymbol{Q}' := \boldsymbol{Q}'' := \boldsymbol{Q}, \quad \boldsymbol{r}'_k := \underline{\boldsymbol{r}}_k, \quad \boldsymbol{r}''_k := \overline{\boldsymbol{r}}_k,$$
$$\boldsymbol{r}'_i := \boldsymbol{r}''_i := \boldsymbol{r}_i \text{ for } i \neq k;$$

compute the estimates $\upsilon' := \Upsilon(\boldsymbol{Q}', \boldsymbol{r}')$ and $\upsilon'' := \Upsilon(\boldsymbol{Q}'', \boldsymbol{r}'')$;

delete the former leading record $(\boldsymbol{Q}, \boldsymbol{r}, \upsilon)$ from the list $\mathcal{L}$;

put the records $(\boldsymbol{Q}', \boldsymbol{r}', \upsilon')$ and $(\boldsymbol{Q}'', \boldsymbol{r}'', \upsilon'')$ into $\mathcal{L}$ so that the values of the third field of the records in $\mathcal{L}$ increases;

denote the first record of the list $\mathcal{L}$ by $(\boldsymbol{Q}, \boldsymbol{r}, \upsilon)$;

`END DO`

$Z := \upsilon$;

summarizes the overall pseudocode of the new algorithm, which we are going to refer to as *parameter partition method* following the terminology tradition of deterministic global optimization [13]. Another suitable name for the new

method is *PPS-method* — after *Partitioning Parameter Set*[1]. The main idea of this kind of method, first presented by the author in [14], can be extended to general nonlinear interval systems of equations, although the result of the subdivision of each interval parameter will be *two subintervals* rather than the endpoints as in the linear case.

If T is the total number of interval (with nonzero widths) elements in the matrix $\boldsymbol{A}$ and right-hand side vector $\boldsymbol{b}$ of the original system (1) (in general, $T \leq (n+1)n$), then the algorithm of Table 1 stops after at most 2^T steps, producing an estimate for $\min\{\, x_\nu \mid x \in \Xi(\boldsymbol{A}, \boldsymbol{b}) \,\}$ from below. How close the computed result is to the exact value of $\min\{\, x_\nu \mid x \in \Xi(\boldsymbol{A}, \boldsymbol{b}) \,\}$ depends mainly on the way we find the estimate $\Upsilon(\boldsymbol{Q}, \boldsymbol{r})$, that is, on the choice of the basic method *Encl*. In particular, for the computed result to be optimal (exactly equal to $\min\{\, x_\nu \mid x \in \Xi(\boldsymbol{A}, \boldsymbol{b}) \,\}$) it is necessary and sufficient that the following condition holds:

the estimate $\Upsilon(Q, r)$ is exact for point linear systems $Qx = r$	(C2)

However, if the dimension of the system under solution is sufficiently large and T exceeds several tens, then, even on modern medium class computers, the simplest parameter partition method will never work till its natural completion, so that it makes good sense to consider it as an *iterative* one.

3 Modifications of Parameter Partition Methods

In this section, we are constructing more sophisticated and, hence, more efficient PPS-methods for the optimal outer estimation of the solution sets to interval linear systems. In doing that, the algorithm of Table 1 shall be a basis to be further improved and modernized by a number of the modifications, some of them being already standard for this kind of algorithm.

3.1 Monotonicity Test

Let an interval linear system $\boldsymbol{Q}x = \boldsymbol{r}$ be given, and we know

$$\frac{\partial x_\nu(\boldsymbol{Q}, \boldsymbol{r})}{\partial q_{ij}} \quad \text{and} \quad \frac{\partial x_\nu(\boldsymbol{Q}, \boldsymbol{r})}{\partial r_i},$$

i.e., interval extensions of the corresponding derivatives

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

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

[1] The more so that there exists a dual class of PSS-methods [7], which exploit the idea of *Partitioning the Solution Set*.

interval $n \times n$-matrix $\tilde{\boldsymbol{Q}} = (\,\tilde{\boldsymbol{q}}_{ij}\,)$ and an interval n-vector $\tilde{\boldsymbol{r}} = (\,\tilde{\boldsymbol{r}}_i\,)$ are formed of the elements

$$\tilde{\boldsymbol{q}}_{ij} = \begin{cases} [\,\underline{\boldsymbol{q}}_{ij}, \underline{\boldsymbol{q}}_{ij}], & \text{for} \quad \dfrac{\partial x_\nu(\boldsymbol{Q},\boldsymbol{r})}{\partial q_{ij}} \geq 0, \\ [\,\overline{\boldsymbol{q}}_{ij}, \overline{\boldsymbol{q}}_{ij}], & \text{for} \quad \dfrac{\partial x_\nu(\boldsymbol{Q},\boldsymbol{r})}{\partial q_{ij}} \leq 0, \\ \boldsymbol{q}_{ij}, & \text{for int} \dfrac{\partial x_\nu(\boldsymbol{Q},\boldsymbol{r})}{\partial q_{ij}} \ni 0, \end{cases} \tag{8}$$

$$\tilde{\boldsymbol{r}}_i = \begin{cases} [\,\underline{\boldsymbol{r}}_i, \underline{\boldsymbol{r}}_i], & \text{for} \quad \dfrac{\partial x_\nu(\boldsymbol{Q},\boldsymbol{r})}{\partial r_i} \geq 0, \\ [\,\overline{\boldsymbol{r}}_i, \overline{\boldsymbol{r}}_i], & \text{for} \quad \dfrac{\partial x_\nu(\boldsymbol{Q},\boldsymbol{r})}{\partial r_i} \leq 0, \\ \boldsymbol{r}_i, & \text{for int} \dfrac{\partial x_\nu(\boldsymbol{Q},\boldsymbol{r})}{\partial r_i} \ni 0, \end{cases} \tag{9}$$

where "int" means interior of the interval, then, evidently,

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

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

We can find the interval extensions of the derivatives entering into the formulas (8)–(9), for instance, in the following way. As is known from any advanced calculus course, the derivatives of the solution x of a linear system $Qx = r$ with respect to its coefficients are given by

$$\frac{\partial x_\nu}{\partial q_{ij}} = -y_{\nu i} x_j, \qquad\qquad \frac{\partial x_\nu}{\partial r_i} = y_{\nu i}$$

providing that $Y = (\,y_{ij})$ is the inverse matrix for $Q = (\,q_{ij})$, $Y = Q^{-1}$ (see e.g. [2], Chapter 16, or [15]). Therefore, if $\boldsymbol{Y} = (\,\boldsymbol{y}_{ij})$ is the so-called "inverse interval matrix" for $\boldsymbol{Q}$, i.e. an enclosure for the set of inverse point matrices, $\boldsymbol{Y} \supseteq \{\, Q^{-1} \mid Q \in \boldsymbol{Q}\,\}$, and $\boldsymbol{x}_j$ is the j-th component of an inclusive interval vector $\boldsymbol{x} \supseteq \Xi(\boldsymbol{Q},\boldsymbol{r})$, we can take the following interval extensions for the derivatives

$$\frac{\partial x_\nu(\boldsymbol{Q},\boldsymbol{r})}{\partial q_{ij}} = -\boldsymbol{y}_{\nu i}\boldsymbol{x}_j, \qquad\qquad \frac{\partial x_\nu(\boldsymbol{Q},\boldsymbol{r})}{\partial r_i} = \boldsymbol{y}_{\nu i}. \tag{10}$$

Computing the "inverse interval matrix" may be carried out, for example, as enclosing of the solution set to the interval matrix equation

$$\boldsymbol{A}Y = I, \qquad I \text{ is the identity matrix,}$$

applying n times (for every column of the matrix Y) the same outer estimation method *Encl* which has been chosen as the basic one for the entire algorithm.

3.2 Subdivision Strategy

Traditionally, the leading intervals are subdivided along the longest components in the interval "branch-and-bound" based global optimization algorithms, which are the nearest relatives to our parameter partition technique. Such a strategy is known to guarantee (see e.g. [12,7]) the convergence of the algorithm, and we also use it in our simplest PPS-method of Table 1 (although it is finite).

When the convergence takes place, we can wish optimizing the numerical procedure, i.e. to achieve the best possible convergence rate, which usually further reduces to the simplified problem of getting faster improvement of the objective function at every step of the algorithm. A strict and exact optimization of the algorithm in the above sense is hardly possible for the parameter partition technique in general, but we are going to improve our method relying on reasonable heuristic considerations and taking into account estimates of the derivatives of the objective function.

If the matrices $\check{Q} = (\,\check{q}_{ij}\,)$ and $\hat{Q} = (\,\hat{q}_{ij}\,)$ differ from each other only in the (k,l)-th entry, $\check{q}_{kl} < \hat{q}_{kl}$, and wid $[\,\check{q}_{kl}, \hat{q}_{kl}]$ stands for the width of the interval $[\,\check{q}_{kl}, \hat{q}_{kl}]$, then, due to Lagrange mean-value theorem,

$$(\hat{Q}^{-1}r)_\nu - (\check{Q}^{-1}r)_\nu = \frac{\partial x_\nu(\tilde{Q}, r)}{\partial q_{kl}} \cdot \text{wid}\,[\,\check{q}_{kl}, \hat{q}_{kl}]$$

for some matrix $\tilde{Q} \in [\,\check{Q}, \hat{Q}]$. Similarly, if the vectors $\check{r} = (\,\check{r}_i)$ and $\hat{r} = (\,\hat{r}_i)$ differ only in the k-th component and $\check{r}_k < \hat{r}_k$, then

$$(Q^{-1}\hat{r})_\nu - (Q^{-1}\check{r})_\nu = \frac{\partial x_\nu(Q, \tilde{r})}{\partial r_k} \cdot \text{wid}\,[\,\check{r}_k, \hat{r}_k]$$

for some vector $\tilde{r} \in [\check{r}, \hat{r}]$.

Now, let the interval matrices $\check{\boldsymbol{Q}}$ and $\hat{\boldsymbol{Q}}$ be obtained from the interval matrix $\boldsymbol{Q}$ by breaking up its element $\boldsymbol{q}_{kl}$ into the endpoints $\underline{\boldsymbol{q}}_{kl}$ and $\overline{\boldsymbol{q}}_{kl}$: $\check{\boldsymbol{q}}_{kl} = \underline{\boldsymbol{q}}_{kl}$, $\hat{\boldsymbol{q}}_{kl} = \overline{\boldsymbol{q}}_{kl}$. Suppose also that $\min\{\, x_\nu \mid x \in \Xi(\check{\boldsymbol{Q}}, \boldsymbol{r})\,\}$ and $\min\{\, x_\nu \mid x \in \Xi(\hat{\boldsymbol{Q}}, \boldsymbol{r})\,\}$ are attained at the same family of the endpoints of the matrix and right-hand side vector, which is almost always the case for "sufficiently narrow" interval systems due to continuity reasons. Then

$$\min\{\, x_\nu \mid x \in \Xi(\hat{\boldsymbol{Q}}, \boldsymbol{r})\,\} - \min\{\, x_\nu \mid x \in \Xi(\check{\boldsymbol{Q}}, \boldsymbol{r})\,\} = \frac{\partial x_\nu(\acute{Q}, \acute{r})}{\partial q_{kl}} \cdot \text{wid}\,\boldsymbol{q}_{kl}$$

for some matrix $\acute{Q} \in \boldsymbol{Q}$ and vector $\acute{r} \in \boldsymbol{r}$. Similarly, let $\check{\boldsymbol{r}}$ and $\hat{\boldsymbol{r}}$ be the interval vectors obtained from the interval vector $\boldsymbol{r}$ by breaking up its k-th component into the endpoints $\underline{\boldsymbol{r}}_k$ and $\overline{\boldsymbol{r}}_k$: $\check{\boldsymbol{r}}_k = \underline{\boldsymbol{r}}_k$, $\hat{\boldsymbol{r}}_k = \overline{\boldsymbol{r}}_k$. Under the condition that $\min\{\, x_\nu \mid x \in \Xi(\boldsymbol{Q}, \check{\boldsymbol{r}})\,\}$ and $\min\{\, x_\nu \mid x \in \Xi(\boldsymbol{Q}, \hat{\boldsymbol{r}})\,\}$ are attained at the same set of the endpoints of the matrix and right-hand side vector, we again get

$$\min\{\, x_\nu \mid x \in \Xi(\boldsymbol{Q}, \hat{\boldsymbol{r}})\,\} - \min\{\, x_\nu \mid x \in \Xi(\boldsymbol{Q}, \check{\boldsymbol{r}})\,\} = \frac{\partial x_\nu(\grave{Q}, \grave{r})}{\partial r_k} \cdot \text{wid}\,\boldsymbol{r}_k$$

for some matrix $\grave{Q} \in \boldsymbol{Q}$ and vector $\grave{r} \in \boldsymbol{r}$. Hence, the value of the product of the width of an interval element by the absolute value of the interval extension of the corresponding derivative may serve as a local measure, in a sense, of how the subdivision of an element from either $\boldsymbol{Q}$ or $\boldsymbol{r}$ affects on $\min\{\, x_\nu \mid x \in \Xi(\boldsymbol{Q},\boldsymbol{r})\,\}$ and the size of the solution set.

For most of the existing techniques that solve interval linear systems, the overestimation of enclosures of the solution sets gets smaller as the solution set lessens. For example, the quadratic convergence is proven for Krawczyk method [5] and so on. Therefore, the decrease of the size of the solution set $\Xi(\boldsymbol{Q},\boldsymbol{r})$ shall result, to approximately the same extent, in the change of the estimate $\Upsilon(\boldsymbol{Q},\boldsymbol{r})$. With such basic methods, the requirement that the objective function should increment most rapidly per step is, in essence, equivalent to that the subdivision of the leading interval system implies the fastest decrease of the size of the solution set.

Taking the above (partly heuristic) conclusions, we thus recommend to subdivide the leading interval systems along the elements on which the maximum of

$$\left|\frac{\partial x_\nu(\boldsymbol{Q},\boldsymbol{r})}{\partial q_{ij}}\right| \cdot \operatorname{wid} \boldsymbol{q}_{ij}, \qquad \left|\frac{\partial x_\nu(\boldsymbol{Q},\boldsymbol{r})}{\partial r_i}\right| \cdot \operatorname{wid} \boldsymbol{r}_i, \tag{11}$$

$i,j = 1,2,\ldots,n$, is attained, that is, along the elements providing the maximal product of the width by the derivative estimate. The author has first proposed such a subdivision strategy requiring maximization of (11) in the paper [14].

3.3 "Rohn Modification"

Beeck-Nickel theorem that we used in Section 2 for the derivation of parameter partition technique has been strengthened in 80s by J. Rohn [6] who defined more precisely the set of the endpoints of the matrix $\boldsymbol{A}$ and right-hand side vector $\boldsymbol{b}$ on which $\min\{\, x_\nu \mid x \in \Xi(\boldsymbol{A},\boldsymbol{b})\,\}$ and $\min\{\, x_\nu \mid x \in \Xi(\boldsymbol{A},\boldsymbol{b})\,\}$ are attained.

To give a mathematically rigorous formulation of the result by Rohn, we need an additional notation. Let

$$\mathcal{E} := \{\, x \in \mathbb{R}^n \mid |x_i| = 1 \ \text{ for } i = 1,2,\ldots,n\,\}$$

be the set of vectors with ± 1 components. For a given interval matrix $\boldsymbol{A}$ and fixed $\sigma,\tau \in \mathcal{E}$, we designate through $A^{\sigma\tau} = (\,a_{ij}^{\sigma\tau}\,)$ a point $n \times n$-matrix formed by the entries

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

Also, we designate through $b^\sigma = (\,b_i^\sigma\,)$ a point n-vector formed by the elements

$$b_i^\sigma := \begin{cases} \overline{\boldsymbol{b}}_i, & \text{if } \ \sigma_i = 1, \\ \underline{\boldsymbol{b}}_i, & \text{if } \ \sigma_i = -1. \end{cases}$$

The matrix $A^{\sigma\tau}$ and the vector b^{σ} are thus made up of collections of the endpoints of the elements of $\boldsymbol{A}$ and $\boldsymbol{b}$ respectively, and there are totally $2^n \cdot 2^n = 4^n$ matrix-vector couples of the form $(A^{\sigma\tau}, b^{\sigma})$ as σ and τ independently vary within $\mathcal{E}$.

For the nonsingular matrix $\boldsymbol{A}$, it turns out [6] that both minimal and maximal component-wise values for the points from the solution set $\Xi(\boldsymbol{A}, \boldsymbol{b})$ can be only reached at the set of 4^n matrices $A^{\sigma\tau}$ and associated vectors b^{σ}, i.e.

$$\min\{\, x_\nu \mid x \in \Xi(\boldsymbol{A}, \boldsymbol{b}) \,\} = \min_{\sigma,\tau \in \mathcal{E}} \left(\left(A^{\sigma\tau} \right)^{-1} b^{\sigma} \right)_\nu , \tag{12}$$

$$\max\{\, x_\nu \mid x \in \Xi(\boldsymbol{A}, \boldsymbol{b}) \,\} = \max_{\sigma,\tau \in \mathcal{E}} \left(\left(A^{\sigma\tau} \right)^{-1} b^{\sigma} \right)_\nu \tag{13}$$

for every index $\nu = 1, 2, \ldots, n$. How could we exploit this fact in our parameter partition method?

It is important to realize that the above result imposes no restriction on the endpoints of a separate element of either $\boldsymbol{A}$ or $\boldsymbol{b}$ unless the information of the other elements' endpoints is drawn into the consideration. The restrictions on the combinations of the endpoints followed from (12)–(13) are essentially collective and to take them into account we should trace the structure of the endpoints involved through all of the matrix $\boldsymbol{A}$ and right-hand side $\boldsymbol{b}$. To put these ideas into practice, we connect, with every interval system $\boldsymbol{Q}x = \boldsymbol{r}$, $\boldsymbol{Q} = (\boldsymbol{q}_{ij})$, $\boldsymbol{r} = (\boldsymbol{r}_i)$, produced from the subdivision of the initial system (1),

1) an auxiliary integer $n \times n$-matrix $W = (w_{ij})$, its elements being equal to ± 1 or 0, such that

$$w_{ij} := \begin{cases} -1, & \text{if } \boldsymbol{q}_{ij} = \overline{\boldsymbol{a}}_{ij}, \\ 0, & \text{if } \boldsymbol{q}_{ij} = \boldsymbol{a}_{ij}, \\ 1, & \text{if } \boldsymbol{q}_{ij} = \underline{\boldsymbol{a}}_{ij}, \end{cases}$$

and

2) auxiliary integer n-vectors $s = (s_i)$ and $t = (t_j)$, their components being equal to ± 1 or 0, such that

$$w_{ij} = s_i t_j \tag{14}$$

for all $i, j = 1, 2, \ldots, n$ and

$$s_i := \begin{cases} -1, & \text{if } \boldsymbol{r}_i = \underline{\boldsymbol{b}}_i, \\ 0, & \text{if } \boldsymbol{r}_i = \boldsymbol{b}_i, \\ 1, & \text{if } \boldsymbol{r}_i = \overline{\boldsymbol{b}}_i. \end{cases}$$

The values of t_j are thus determined implicitly through the matrix W and vector s. Additionally, the working list $\mathcal{L}$ is going to consist of the records with *six* members,

$$\Big(\boldsymbol{Q},\ \boldsymbol{r},\ \Upsilon(\boldsymbol{Q}, \boldsymbol{r}),\ W,\ s,\ t \Big), \tag{15}$$

to preserve W, s and t obtained at the preceding steps of the algorithm.

In the rest of the paper, we will call W and s, t *check matrix* and *check vectors* respectively, intending to use them for checking and controlling the overall

bisection process in the PPS-methods. Namely, the vectors s and t are to be "approximations" to the vectors σ and τ, respectively, from the equalities (12)–(13), while $W = s\,t^\top$ shall be an "approximation" to the matrix $(\sigma\,\tau^\top)$. At the start of our algorithm, we set W, s and t to all zeros, and then they are recalculated (updated) so as to replace their zero elements (that correspond to our ignorance of which specific endpoint is treated) to nonzero ones (that correspond to a certain endpoint). The check matrix W and check vectors s, t, mutually affecting each other and being updated during the algorithm run, are thus intended to "supervise" the partitioning of the initial interval linear system so that only the variants allowed by the equalities (12)–(13) are begotten. The latter is implemented as follows.

At each step of the algorithm, when subdividing an interval element $\boldsymbol{h}$ of the leading system $\boldsymbol{Q}x = \boldsymbol{r}$, we look at the corresponding value

- of the check matrix $W = (\,w_{ij})$, if the element $\boldsymbol{h}$ is $\boldsymbol{q}_{kl}$ of the matrix $\boldsymbol{Q}$,
- of the check vector $s = (s_i)$, if the element $\boldsymbol{h}$ is $\boldsymbol{r}_k$ of the right-hand side vector $\boldsymbol{r}$.

Then, in case of $w_{kl} = 0$ ($s_k = 0$ respectively), we engender, according to the usual subdivision procedure used in the simplest parameter partition method of Table 1, *two* interval systems-descendants $\boldsymbol{Q}'x = \boldsymbol{r}'$ and $\boldsymbol{Q}''x = \boldsymbol{r}''$ corresponding to both endpoints of the interval subdivided. Otherwise, in case of $w_{kl} \neq 0$ ($s_k \neq 0$ respectively), only *one* descendant $\boldsymbol{Q}'x = \boldsymbol{r}'$ may be engendered, depending on the sign of w_{kl} (s_k respectively). More precisely, we perform the procedure presented in Table 2 instead of the traditional bisection.

Why is that at all possible? In other words, can discarding the second interval system-descendant in the above procedure violate the fact that the leading estimate $\Upsilon(\boldsymbol{Q}, \boldsymbol{r})$ approximates the sought-for $\min\{\, x_\nu \mid x \in \Xi(\boldsymbol{A}, \boldsymbol{b})\,\}$ from below?

To answer these questions, we note that in the new subdivision procedure of Table 2 we reject only such interval systems that neither belong to the set of point systems

$$\{\, A^{\sigma\tau}x = b^\sigma \mid \sigma, \tau \in \mathcal{E}\,\}$$

nor contain them. Therefore, due to the property (C1) of the basic enclosing method and taking into account the equality (12), we have

$$\begin{aligned}
\min\{\, x_\nu \mid x \in \Xi(\boldsymbol{A}, \boldsymbol{b})\,\} \;&=\; \min_{\sigma,\tau\in\mathcal{E}} \Big(\big(A^{\sigma\tau}\big)^{-1} b^\sigma \Big)_\nu \;\geq\; \min_{\sigma,\tau\in\mathcal{E}} \Upsilon(A^{\sigma\tau}, b^\sigma) \\
&\geq\; \min\{\, \Upsilon(\boldsymbol{Q}, \boldsymbol{r}) \mid\; \boldsymbol{Q} \ni A^{\sigma\tau} \text{ and } \boldsymbol{r} \ni b^\sigma \text{ for some } \sigma, \tau \in \mathcal{E}\,\} \\
&\geq\; \min\{\, \Upsilon(\boldsymbol{Q}, \boldsymbol{r}) \mid\; \text{the system } \boldsymbol{Q}x = \boldsymbol{r} \text{ is in the working list } \mathcal{L}\,\} \\
&=\; \text{the leading estimate } \Upsilon(\boldsymbol{Q}, \boldsymbol{r}),
\end{aligned}$$

so that with our new subdivision procedure the leading estimate really approximates $\min\{\, x_\nu \mid x \in \Xi(\boldsymbol{A}, \boldsymbol{b})\,\}$ from below.

Table 2. Generating the systems-descendants

IF (subdivided is $\boldsymbol{q}_{kl}$) THEN

IF ($w_{kl} = 0$) THEN

generate the systems $\boldsymbol{Q}'x = \boldsymbol{r}'$ and $\boldsymbol{Q}''x = \boldsymbol{r}''$ so that

$$\boldsymbol{q}'_{ij} := \boldsymbol{q}''_{ij} := \boldsymbol{q}_{ij} \quad \text{for } (i,j) \neq (k,l),$$

$$\boldsymbol{q}'_{kl} := \underline{\boldsymbol{q}}_{kl}, \quad \boldsymbol{q}''_{kl} := \overline{\boldsymbol{q}}_{kl}, \qquad \boldsymbol{r}' := \boldsymbol{r}'' := \boldsymbol{r};$$

ELSE

generate the system $\boldsymbol{Q}'x = \boldsymbol{r}'$ so that

$$\boldsymbol{r}' := \boldsymbol{r}, \qquad \boldsymbol{q}'_{ij} := \boldsymbol{q}_{ij} \quad \text{for } (i,j) \neq (k,l),$$

$$\boldsymbol{q}'_{kl} := \begin{cases} \underline{\boldsymbol{q}}_{kl}, & \text{for } w_{kl} = 1, \\ \overline{\boldsymbol{q}}_{kl}, & \text{for } w_{kl} = -1; \end{cases}$$

END IF

END IF

IF (subdivided is $\boldsymbol{r}_k$) THEN

IF ($s_k = 0$) THEN

generate the systems $\boldsymbol{Q}'x = \boldsymbol{r}'$ and $\boldsymbol{Q}''x = \boldsymbol{r}''$ so that

$$\boldsymbol{Q}' := \boldsymbol{Q}'' := \boldsymbol{Q},$$

$$\boldsymbol{r}'_i := \boldsymbol{r}_i \quad \text{for } i \neq k, \qquad \boldsymbol{r}'_k := \underline{\boldsymbol{r}}_k, \quad \boldsymbol{r}''_k := \overline{\boldsymbol{r}}_k;$$

ELSE

generate the system $\boldsymbol{Q}'x = \boldsymbol{r}'$ so that

$$\boldsymbol{Q}' := \boldsymbol{Q}, \qquad \boldsymbol{r}'_i := \boldsymbol{r}_i \quad \text{for } i \neq k,$$

$$\boldsymbol{r}'_k := \begin{cases} \underline{\boldsymbol{r}}_k, & \text{for } s_k = -1, \\ \overline{\boldsymbol{r}}_k, & \text{for } s_k = 1; \end{cases}$$

END IF

END IF

Getting started to specify the formal computational scheme, let us establish the rules for the recalculation of the check matrix W and check vectors s, t during the algorithm run. In doing this, we adopt the following notation: if a leading interval system $\boldsymbol{Q}x = \boldsymbol{r}$ has begotten, as the result of executing the algorithm of Table 2, the systems-descendants $\boldsymbol{Q}'x = \boldsymbol{r}'$ and $\boldsymbol{Q}''x = \boldsymbol{r}''$, then the corresponding new check matrices and check vectors will be referred to as W', W'' and s', s'', t', t'' respectively. There exists a one-to-one correspondence between the vector s and the right-hand side of the interval system $\boldsymbol{Q}x = \boldsymbol{r}$, while partitioning the interval matrix $\boldsymbol{Q}$ of the system affects the vectors s and t only implicitly, through the matrix W and the conditions (14). The latter still enables us to organize recalculating of W, s and t at every such algorithm step that it results in the subdivision of an interval element of the leading system. Otherwise,

if the leading interval system engenders only one descendant $\boldsymbol{Q}'x = \boldsymbol{r}'$ according to Table 2, the check vectors s, t and check matrix W remain unchanged so that $s' := s$, $t' := t$, $W' := W$.

So, let the leading interval system $\boldsymbol{Q}x = \boldsymbol{r}$ have been subdivided to two systems-descendants $\boldsymbol{Q}'x = \boldsymbol{r}'$ and $\boldsymbol{Q}''x = \boldsymbol{r}''$ defined as in Table 2. What should be the law according to which we are to form the matrices W', W'' and vectors s', s'', t', t'' corresponding to the systems-descendants? Initially, we can set $W'' := W' := W$, $s'' := s' := s$, $t'' := t' := t$, and then perform the following two-stage recalculating procedure:

First, we modify W', W'' and s', s'' using the information about the subdivision done. Namely,

(i) if the subdivided element was $\boldsymbol{q}_{kl}$ of the matrix $\boldsymbol{Q}$, then, in the matrices $W' = (w'_{ij})$ and $W'' = (w''_{ij})$, we put $w'_{kl} := 1$ and $w''_{kl} := -1$;
(ii) if the subdivided element was $\boldsymbol{r}_k$ of the right-hand side vector $\boldsymbol{r}$, then, in the vectors $s' = (s'_i)$ and $s'' = (s''_i)$, we put $s'_k := -1$ and $s''_k := 1$.

Second, we recalculate each of the two families of the interconnected objects — W', s', t' and W'', s'', t'' respectively — using the relations (14). Namely,

(i) if s' or t' is changed, we try to update the matrix W';
(ii) if W' or t' is changed, we try to update the vector s';
(iii) if W' or s' is changed, we try to update the vector t'.

The instructions (i)–(iii) repeat consecutively one after another in a cycle until changes in W', s' and t' stop. The same process is then applied to W'', s'', t''.

The overall algorithmic scheme of the above procedure turns out to be quite involved so that it makes sense to provide the reader with its more strict and detailed description. Table 3 presents the corresponding pseudocode and some explanations are in order. The Boolean variables

$$W'Change, \quad s'Change, \quad t'Change, \quad W'C, \quad s'C, \quad t'C$$

and

$$W''Change, \quad s''Change, \quad t''Change, \quad W''C, \quad s''C, \quad t''C$$

are "flags" introduced to reflect the current state of changes in the check matrices and vectors W', s', t' and W'', s'', t'' respectively. The value `true` means that the corresponding object has been changed at the current iteration of the recalculating process, while the value `false` means "no changes". The whole algorithm of Table 3 can be divided into three essentially different parts. The first one, consisting of two starting lines, is preparatory and represents initialization of the flags. The second part, consisting of two conditional operators `IF-THEN`, fulfills recalculation of the check matrices W', W'' and check vectors s', s'' taking into account the bisection results. Finally, the third part of the pseudocode, consisting

Table 3. Recalculation of W', W'', s', s'', t', t''

```
W'Change := false;      s'Change := false;      t'Change := false;
W''Change := false;     s''Change := false;     t''Change := false;
IF ( (subdivided is q_kl of Q) AND (q_kl is subdivided to two endpoints) ) THEN
     w'_kl := 1;  w''_kl := −1;   W'Change := true;   W''Change := true;
END IF
IF ( (subdivided is r_k of r) AND (r_k is subdivided to two endpoints) ) THEN
     s'_k := −1;  s''_k := 1;   s'Change := true;   s''Change := true;
END IF
DO WHILE ( W'Change OR s'Change OR t'Change )
     IF ( s'Change OR t'Change ) THEN
          trying to update the matrix W' according to (14);
          IF ( W' has been changed ) THEN W'C := true
          ELSE W'C := false END IF
     END IF
     IF ( W'Change OR t'Change ) THEN
          trying to update the vector s' according to (14);
          IF ( s' has been changed ) THEN s'C := true
          ELSE s'C := false END IF
     END IF
     IF ( W'Change OR s'Change ) THEN
          trying to update the vector t' according to (14);
          IF ( t' has been changed ) THEN t'C := true
          ELSE t'C := false END IF
     END IF
     W'Change := W'C;  s'Change := s'C;  t'Change := t'C;
END DO
DO WHILE ( W''Change OR s''Change OR t''Change )
     IF ( s''Change OR t''Change ) THEN
          trying to update the matrix W'' according to (14);
          IF ( W'' has been changed ) THEN W''C := true
          ELSE W''C := false END IF
     END IF
     IF ( W''Change OR t''Change ) THEN
          trying to update the vector s'' according to (14);
          IF ( s'' has been changed ) THEN s''C := true
          ELSE s''C := false END IF
     END IF
     IF ( W''Change OR s''Change ) THEN
          trying to update the vector t'' according to (14);
          IF ( t'' has been changed ) THEN t''C := true
          ELSE t''C := false END IF
     END IF
     W''Change := W''C;  s''Change := s''C;  t''Change := t'C;
END DO
```

of two DO WHILE cycles, makes an attempt to update the new check matrices and check vectors "on its own basis", according to the main relation (14). We perform the calculation until W', W'', s', s'', t', t'' "stabilizes", that is, their changes stop and all the corresponding flags $W'Change$, $s'Change$, $t'Change$, $W'C$, $s'C$, $t'C$, $W''Change$, $s''Change$, $t''Change$, $W''C$, $s''C$, $t''C$ are false.

To complete the formalized description of "Rohn modification" we have only to define in detail what is meant by "trying to update the matrix W' according to (14)", "trying to update the vector s'" and the like in Table 3.

Let us denote by Greek capital letters K', Λ' and Ω' index subsets of the elements of the vector s', vector t' and the matrix W' respectively that have changed their values (from 0 to ± 1) at the current step of the recalculation procedure of Table 3. K' and Λ' are thus subsets of the set of the first n natural numbers $\{1, 2, \ldots, n\}$, while Ω is a subset of the set of all the pairs constituted by the first n natural numbers, i.e. of $\{(1,1),(1,2),\ldots,(1,n),(2,1),\ldots,(n,n)\}$. Each of the sets K', Λ', Ω' may be empty, or may contain more than one member. Then our "trying to update the vector s'" may be organized as follows:

Table 4. Updating s'

```
IF ( W'Change ) THEN
     DO FOR  (k,l) ∈ Ω'
          IF ( t'_l ≠ 0 ) s'_k := w'_kl / t'_l
     END DO
END IF
IF ( t'Change ) THEN
     DO FOR  l ∈ Λ'
          DO FOR  k = 1  TO  n
               IF ( s_k = 0 AND w'_kl ≠ 0 ) s'_k := w'_kl / t'_l
          END DO
     END DO
END IF
```

"Trying to update the vector t'" can be accomplished similar to the above with the only distinction that the cycle "DO FOR $l \in \Lambda'$" in the second IF-operator should be replaced by "DO FOR $k \in K'$". Coming up next is the updating of W': The same with the recalculation of s'', t'' and W'', for which we should introduce the index subsets K'', Λ'' and Ω'' to denote the indices of the elements of the vector s'', vector t'' and matrix W'' respectively that has changed at the current step of the recalculation process.

Finally, it is worth mentioning the following remarkable property of the check matrix W: in every its 2×2-submatrix, any element is equal to the product of the rest three elements. To make sure of that, designate by i_1, i_2 the numbers of the rows and by j_1, j_2 the numbers of the columns forming the submatrix under consideration. Then, according to the definition (14).

Table 5. Updating W'

```
IF ( s'Change ) THEN
     DO FOR  k ∈ K'
          DO FOR  l = 1  TO  n
               IF ( t'_l ≠ 0 ) w'_kl := s'_k t'_l
          END DO
     END DO
END IF
IF ( t'Change ) THEN
     DO FOR  l ∈ Λ'
          DO FOR  k = 1  TO  n
               IF ( s'_k ≠ 0 ) w'_kl := s'_k t'_l
          END DO
     END DO
END IF
```

$$w_{i_1 j_1} = \sigma_{i_1}\tau_{j_1}, \qquad w_{i_1 j_2} = \sigma_{i_1}\tau_{j_2},$$
$$w_{i_2 j_1} = \sigma_{i_2}\tau_{j_1}, \qquad w_{i_2 j_2} = \sigma_{i_2}\tau_{j_2}.$$

Multiplying any three of the above equalities, e.g., the 1st, 2nd and 4th, we get

$$w_{i_1 j_1} w_{i_1 j_2} w_{i_2 j_2} = \sigma_{i_1}\tau_{j_1}\sigma_{i_1}\tau_{j_2}\sigma_{i_2}\tau_{j_2}.$$

The square of any of the components of σ and τ is 1, so that

$$w_{i_1 j_1} w_{i_1 j_2} w_{i_2 j_2} = \tau_{j_1}\sigma_{i_2} = w_{i_2 j_1}. \tag{16}$$

The same with the rest elements of the submatrix:

$$w_{i_1 j_1} w_{i_1 j_2} w_{i_2 j_1} = w_{i_2 j_2}, \tag{17}$$
$$w_{i_1 j_2} w_{i_2 j_1} w_{i_2 j_2} = w_{i_1 j_1}, \tag{18}$$
$$w_{i_1 j_1} w_{i_2 j_1} w_{i_2 j_2} = w_{i_1 j_2}. \tag{19}$$

Sometimes, the relations (16)–(19) may prove helpful in further refining the check matrix W. For instance, let us be about to subdivide the leading interval system $\boldsymbol{Q}x = \boldsymbol{r}$ in the element $\boldsymbol{q}_{kl}$ while the corresponding element w_{kl} of the check matrix W is zero, i.e. normally we will have to engender two systems-descendants instead of $\boldsymbol{Q}x = \boldsymbol{r}$. It is advisable to make an effort to determine w_{kl} by searching 2×2-submatrices of W having all the entries nonzero except w_{kl}. If such a submatrix in W is found, we assign w_{kl} the product of the rest its three elements. The above stated can be implemented as the following program.

Table 6. Refining W by 2×2-submatrix search

```
DO FOR  i = 1  TO  n
     DO FOR  j = 1  TO  n
          IF ( i ≠ k AND j ≠ l ) THEN
               IF ( w_ij ≠ 0  AND  w_il ≠ 0  AND  w_kj ≠ 0 ) THEN
                    w_kl := w_ij w_il w_kj;
                    EXIT
               END IF
          END IF
     END DO
END DO
```

where `EXIT` operator means leaving all the blocks and cycles in the above code.

3.4 Sifting Unpromising Records

Next, we consider the modification of the parameter partition method resulting from the computation of the estimates Υ for the midpoints of the leading systems. It enables us to partially control the precision of the current estimate for $\min\{\, x_\nu \mid x \in \Xi(\boldsymbol{A}, \boldsymbol{b}), \}$, as well as to delete *unpromising records*, which never become the leading ones, from the working list $\mathcal{L}$. Thanks to the last feature the growth of the list $\mathcal{L}$ is confined to some extent.

Let, along with the estimates $\Upsilon(\boldsymbol{Q}, \boldsymbol{r})$ for the interval linear systems $\boldsymbol{Q}x = \boldsymbol{r}$, the values $\Upsilon(\boxdot\boldsymbol{Q}, \boxdot\boldsymbol{r})$ be computed during the algorithm run where "$\boxdot$" means taking a point from the interval. It is fairly simple to realize that

$$\Upsilon(\boxdot\boldsymbol{Q}, \boxdot\boldsymbol{r}) \geq \Upsilon(\boldsymbol{Q}, \boldsymbol{r})$$

and the values of $\Upsilon(\boxdot\boldsymbol{Q}, \boxdot\boldsymbol{r})$ approximate $\min\{\, x_\nu \mid x \in \Xi(\boldsymbol{A}, \boldsymbol{b}) \,\}$ from above. If we define, for each step of our PPS-method,

$$\omega := \min \Upsilon(\boxdot\boldsymbol{Q}, \boxdot\boldsymbol{r}) \tag{20}$$

over all the interval linear systems $\boldsymbol{Q}x = \boldsymbol{r}$ which have been in the list $\mathcal{L}$ up to the current step, then

$$\min\{\, x_\nu \mid x \in \Xi(\boldsymbol{A}, \boldsymbol{b}) \,\} \leq \omega.$$

On the other hand, if $\boldsymbol{Q}x = \boldsymbol{r}$ is the leading interval system, then

$$\Upsilon(\boldsymbol{Q}, \boldsymbol{r}) \leq \min\{\, x_\nu \mid x \in \Xi(\boldsymbol{A}, \boldsymbol{b}) \,\},$$

so that one more stopping criterion in our algorithm may be attaining the required smallness of $(\omega - \Upsilon(\boldsymbol{Q}, \boldsymbol{r}))$.

Next, an interval linear system $\boldsymbol{Q}x = \boldsymbol{r}$ satisfying at some step the condition

$$\Upsilon(\boldsymbol{Q}, \boldsymbol{r}) > \omega \tag{21}$$

will never become the leading one, and deleting the corresponding record from the working list $\mathcal{L}$ would have no effect on the result of the execution of the algorithm. In general, all the newly generated records should be tested by the inequality (21), while the total clean up of the working list — looking through $\mathcal{L}$ and removing the records satisfying (21) from it — makes sense only after the value of ω changes.

Choosing $(\boxdot\boldsymbol{Q}, \boxdot\boldsymbol{r}) \in \operatorname{Arg}\ \min\{\, \Upsilon(Q,r) \mid Q \in \boldsymbol{Q}, r \in \boldsymbol{r}\,\}$ would be an ideal outcome, but in general it is not at all easier than the original problem (3)–(4). So, to minimize the possible deviation of $(\boxdot\boldsymbol{Q}, \boxdot\boldsymbol{r})$ from the set $\operatorname{Arg}\min\{\, \Upsilon(Q,r) \mid Q \in \boldsymbol{Q}, r \in \boldsymbol{r}\,\}$, we can take $\boxdot\boldsymbol{Q}$ and $\boxdot\boldsymbol{r}$ as the midpoints of the matrix $\boldsymbol{Q}$ and right-hand side vector $\boldsymbol{r}$, i.e. as $\operatorname{mid}\boldsymbol{Q}$ and $\operatorname{mid}\boldsymbol{r}$ respectively.

3.5 Influence of the Basic Method

To start many procedures for enclosing the solution sets to interval linear systems, we need an initial interval that contains the solution set under estimation. Such are, for instance, interval Gauss-Seidel iteration, Krawczyk method and some others. It is not hard to understand that an enclosure of the solution set to a leading system $\boldsymbol{Q}x = \boldsymbol{r}$, found at a previous step, may serve as an initial approximation for the procedures enclosing the solution sets to the systems-descendants $\boldsymbol{Q}'x = \boldsymbol{r}'$ with $\boldsymbol{Q}' \subseteq \boldsymbol{Q}$ and $\boldsymbol{r}' \subseteq \boldsymbol{r}$. The same trick is applicable to the computation of the "inverse interval matrix" which we need in the monotonicity test of §3.1 and in selecting the subdivided element according to the technique of §3.2.

Hence, having chosen such a basic method that requires a starting outer approximation, it makes sense to preserve the interval enclosures of both the solution set and "inverse interval matrix" obtained at the preceding step of the algorithm. To do that, we have to enlarge the records forming the working list $\mathcal{L}$ by two more fields, so that now $\mathcal{L}$ contains neither triples (7) nor six-term records (15), but eight-term records

$$\Big(\ \boldsymbol{Q},\ \boldsymbol{r},\ \Upsilon(\boldsymbol{Q},\ \boldsymbol{r}),\ W,\ s,\ t,\ \boldsymbol{Y},\ \boldsymbol{x}\ \Big),$$

where the first three fields has the same meaning as in (7), the check matrix W and check vectors s, t have been introduced in §3.3 and, additionally,

$\boldsymbol{Y}$ is an interval $n \times n$-matrix, such that $\boldsymbol{Y} \supseteq \{\, Q^{-1} \mid Q \in \boldsymbol{Q}\,\}$,
$\boldsymbol{x}$ is an interval n-vector, such that $\boldsymbol{x} \supseteq \Xi(\boldsymbol{Q}, \boldsymbol{r})$.

Every technique that encloses the solution set to interval linear systems and satisfies the condition (C2) usually produces the exact estimate $\Upsilon(\boldsymbol{Q}, \boldsymbol{r})$ not only for point (noninterval) $\boldsymbol{Q}$ and $\boldsymbol{r}$, but for a wider data set, when a part of the elements in either $\boldsymbol{Q}$ or $\boldsymbol{r}$ can be intervals. For instance, interval Gauss-Seidel iteration [4,5] and stationary iterative single-step and total-step procedures [2] provide the exact estimate $\Upsilon(\boldsymbol{Q}, \boldsymbol{r})$ in case of the point matrix $\boldsymbol{Q}$, no matter what the right-hand side vector $\boldsymbol{r}$ is. The estimate $\Upsilon(\boldsymbol{Q}, \boldsymbol{r})$ obtained through interval Gauss elimination is, obviously, exact for the triangular matrices $\boldsymbol{Q}$. The list

of examples might be continued as well. There can exist more sophisticated conditions on the mutual disposition of the elements in the interval matrix $\boldsymbol{Q}$ and vector $\boldsymbol{r}$, their widths, magnitudes, and so on.

We thus may not wait for the complete "deintervalization" of the leading interval system $\boldsymbol{Q}x = \boldsymbol{r}$ (the termination criteria for "`DO WHILE`" cycle in Table 1) to stop the PPS-method. Instead, it is quite sufficient that the leading estimate $\Upsilon(\boldsymbol{Q}, \boldsymbol{r})$ is exact.

One can go even further and make provision for a dynamic runtime change of the basic method *Encl*. Originally, *Encl* may be a technique with a wide applicability scope, although having a low convergence rate. Afterward, as the algorithm reaches a prescribed narrowness of the interval systems-descendants, we can switch *Encl* to a more precise specialized technique.

The conclusion one ought to draw from the above stated is that, to achieve the best possible efficacy of the parameter partition methods, all the components of the practical algorithm, namely

data structure (in particular, the form of the records from $\mathcal{L}$),
the way the working list $\mathcal{L}$ is processed,
subdivision strategy, etc.,

should be carefully adapted to the features of the specific problem under solution.

3.6 Overall Computational Scheme

The pseudocodes of Tables 7 and 8 sum up the above modifications of the parameter partition method for outer interval estimation of the solution sets to the interval linear systems.

Theoretically, Rohn modification enables us to decrease the upper bound of the computational complexity of PPS-methods from 2^{n^2+n} to 4^n, but

- This is done at the price of substantial complication of the algorithm, so that its informational complexity becomes quite high,
- When solving large practical problems of the dimensions greater than several tens, both 2^{n^2+n} and 4^n are unattainable and parameter partition technique should be considered rather as an iteration procedure that does not work to the natural completion. As a consequence, the check matrix W remains mostly zero and we cannot avail ourselves of the information followed from the equalities (12)–(13).

To our mind, it is up to the user to decide in each specific case, judging on the size of the interval system, its structure, etc., whether incorporating Rohn modification into this or that implementation of parameter partition technique is really expedient. This is why we present two overall computational schemes, both with and without Rohn modification.

In Tables 7 and 8, it is supposed that interval Gauss-Seidel iteration is taken as the basic enclosing method *Encl*, but this is set just for certainty. We should emphasize that the parameter partition method is rather a general scheme, while Tables 7 and 8 present only some of its possible implementations. The constructions of the above subsections contain, in particular, a number of "free variables"

Table 7. Algorithm `LinPPS1`

`DO WHILE` $\Big($ (the leading estimate $\Upsilon(\boldsymbol{Q},\boldsymbol{r})$ is not exact) `OR` $(\omega - \Upsilon(\boldsymbol{Q},\boldsymbol{r}) > \epsilon)\Big)$

using the formulas (10), compute the interval enclosures of the derivatives

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

that correspond to the interval elements $\boldsymbol{q}_{ij}$ and $\boldsymbol{r}_i$ with nonzero width;

"squeeze" according to (8)–(9) the interval elements of $\boldsymbol{Q}$ and $\boldsymbol{r}$ for which we have detected the monotonicity of x_ν with respect to q_{ij} and r_j, denote the interval matrix and vector thus obtained by $\boldsymbol{Q}$ and $\boldsymbol{r}$ too;

find, among the elements of the system $\boldsymbol{Q}x = \boldsymbol{r}$, an interval $\boldsymbol{h}$ that corresponds to the largest of the products

$$\left|\frac{\partial x_\nu(\boldsymbol{Q},\boldsymbol{r})}{\partial q_{ij}}\right| \cdot \operatorname{wid} \boldsymbol{q}_{ij}, \quad \left|\frac{\partial x_\nu(\boldsymbol{Q},\boldsymbol{r})}{\partial r_i}\right| \cdot \operatorname{wid} \boldsymbol{r}_i, \quad i,j \in \{1,2,\ldots,n\};$$

beget the interval "systems-descendants" $\boldsymbol{Q}'x = \boldsymbol{r}'$ and $\boldsymbol{Q}''x = \boldsymbol{r}''$ so that
if $\boldsymbol{h} = \boldsymbol{q}_{kl}$ for some $k,l \in \{1,2,\ldots,n\}$, then set
$\boldsymbol{q}'_{ij} := \boldsymbol{q}''_{ij} := \boldsymbol{q}_{ij}$ for $(i,j) \neq (k,l)$, $\quad \boldsymbol{q}'_{kl} := \underline{\boldsymbol{q}}_{kl}$, $\boldsymbol{q}''_{kl} := \overline{\boldsymbol{q}}_{kl}$,
$\boldsymbol{r}' := \boldsymbol{r}'' := \boldsymbol{r}$;
if $\boldsymbol{h} = \boldsymbol{r}_k$ for some $k \in \{1,2,\ldots,n\}$, then set
$\boldsymbol{r}'_i := \boldsymbol{r}''_i := \boldsymbol{r}_i$ for $i \neq k$, $\quad \boldsymbol{r}'_k := \underline{\boldsymbol{r}}_k$, $\boldsymbol{r}''_k := \overline{\boldsymbol{r}}_k$,
$\boldsymbol{Q}' := \boldsymbol{Q}'' := \boldsymbol{Q}$;

compute the interval vectors $\boldsymbol{x}' = \mathit{Encl}\,(\boldsymbol{Q}',\boldsymbol{r}')$ and $\boldsymbol{x}'' = \mathit{Encl}\,(\boldsymbol{Q}'',\boldsymbol{r}'')$, taking $\boldsymbol{x}$ as an initial approximation;

assign the estimates $\upsilon' := \Upsilon(\boldsymbol{Q}',\boldsymbol{r}')$ and $\upsilon'' := \Upsilon(\boldsymbol{Q}'',\boldsymbol{r}'')$;

sharpen the enclosures for the "inverse interval matrices" $\boldsymbol{Y}' \supseteq (\boldsymbol{Q}')^{-1}$ and $\boldsymbol{Y}'' \supseteq (\boldsymbol{Q}'')^{-1}$, taking $\boldsymbol{Y}$ as an initial approximation;

compute the estimates $\Upsilon(\operatorname{mid}\boldsymbol{Q}', \operatorname{mid}\boldsymbol{r}')$ and $\Upsilon(\operatorname{mid}\boldsymbol{Q}'', \operatorname{mid}\boldsymbol{r}'')$, and set
$\mu := \min\{\Upsilon(\operatorname{mid}\boldsymbol{Q}', \operatorname{mid}\boldsymbol{r}'), \Upsilon(\operatorname{mid}\boldsymbol{Q}'', \operatorname{mid}\boldsymbol{r}'')\}$;

delete the former leading record $(\boldsymbol{Q},\boldsymbol{r},\upsilon,\boldsymbol{Y},\boldsymbol{x})$ from the list $\mathcal{L}$;

if $\upsilon' \leq \omega$, then put the record $(\boldsymbol{Q}',\boldsymbol{r}',\upsilon',\boldsymbol{Y}',\boldsymbol{x}')$ into the list $\mathcal{L}$ so that the values of the third field of the records in $\mathcal{L}$ increase;

if $\upsilon'' \leq \omega$, then put the record $(\boldsymbol{Q}'',\boldsymbol{r}'',\upsilon'',\boldsymbol{Y}'',\boldsymbol{x}'')$ into the list $\mathcal{L}$ so that the values of the third field of the records in $\mathcal{L}$ increase;

if $\omega > \mu$, then set $\omega := \mu$ and clean up the list $\mathcal{L}$, i.e.
remove from it all such records $(\boldsymbol{Q},\boldsymbol{r},\upsilon,\boldsymbol{Y},\boldsymbol{x})$, that $\upsilon > \omega$;

`END DO`

Table 8. Algorithm `LinPPS2`

`DO WHILE` $\Big($ (the leading estimate $\Upsilon(\boldsymbol{Q},\boldsymbol{r})$ is not exact) `OR` $(\omega - \Upsilon(\boldsymbol{Q},\boldsymbol{r}) > \epsilon)\Big)$

using the formulas (10), compute the interval enclosures of the derivatives

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

that correspond to the interval elements $\boldsymbol{q}_{ij}$ and $\boldsymbol{r}_i$ with nonzero width;

"squeeze" according to (8)–(9) the interval elements of $\boldsymbol{Q}$ and $\boldsymbol{r}$ for which we have detected the monotonicity of x_ν with respect to q_{ij} and r_j, denote the interval matrix and vector thus obtained by $\boldsymbol{Q}$ and $\boldsymbol{r}$ too;

find, among the elements of the system $\boldsymbol{Q}x = \boldsymbol{r}$, an interval $\boldsymbol{h}$ that corresponds to the largest of the products

$$\left|\frac{\partial x_\nu(\boldsymbol{Q},\boldsymbol{r})}{\partial q_{ij}}\right| \cdot \operatorname{wid}\boldsymbol{q}_{ij}, \quad \left|\frac{\partial x_\nu(\boldsymbol{Q},\boldsymbol{r})}{\partial r_i}\right| \cdot \operatorname{wid}\boldsymbol{r}_i, \quad i,j \in \{1,2,\ldots,n\};$$

trying to refine the check matrix W according to the procedure of Table 6;

beget one or two interval "systems-descendants" $\boldsymbol{Q}'x = \boldsymbol{r}'$ and $\boldsymbol{Q}''x = \boldsymbol{r}''$ according to the procedure of Table 2;

if *two* systems-descendants have been generated, calculate the new check matrices W', W'' and vectors s', s'', t', t'' according to the procedures of Table 3 and Tables 4–5; otherwise, set $W' := W$, $s' := s$, $t' := t$;

compute the interval vectors $\boldsymbol{x}' = \mathit{Encl}\,(\boldsymbol{Q}',\boldsymbol{r}')$ and, possibly, $\boldsymbol{x}'' = \mathit{Encl}\,(\boldsymbol{Q}'',\boldsymbol{r}'')$, taking $\boldsymbol{x}$ as the initial approximation;

assign the estimates $\upsilon' := \Upsilon(\boldsymbol{Q}',\boldsymbol{r}')$ and, possibly, $\upsilon'' := \Upsilon(\boldsymbol{Q}'',\boldsymbol{r}'')$;

sharpen the enclosures for the "inverse interval matrices" $\boldsymbol{Y}' \supseteq (\boldsymbol{Q}')^{-1}$ and, possibly, $\boldsymbol{Y}'' \supseteq (\boldsymbol{Q}'')^{-1}$, taking $\boldsymbol{Y}$ as the initial approximation;

compute the estimates $\Upsilon(\operatorname{mid}\boldsymbol{Q}',\operatorname{mid}\boldsymbol{r}')$ and, possibly, $\Upsilon(\operatorname{mid}\boldsymbol{Q}'',\operatorname{mid}\boldsymbol{r}'')$, and set $\mu := \min\{\Upsilon(\operatorname{mid}\boldsymbol{Q}',\operatorname{mid}\boldsymbol{r}'),\Upsilon(\operatorname{mid}\boldsymbol{Q}'',\operatorname{mid}\boldsymbol{r}'')\}$;

delete the former leading record $(\boldsymbol{Q},\boldsymbol{r},\upsilon,\boldsymbol{Y},W,s,t,\boldsymbol{x})$ from the list $\mathcal{L}$;

if $\upsilon' \le \omega$, then put the record $(\boldsymbol{Q}',\boldsymbol{r}',\upsilon',W',s',t',\boldsymbol{Y}',\boldsymbol{x}')$ into the list $\mathcal{L}$ so that the values of the third field of the records in $\mathcal{L}$ increase;

if the two systems-descendants has been engendered and $\upsilon'' \le \omega$, then put the record $(\boldsymbol{Q}'',\boldsymbol{r}'',\upsilon'',W'',s'',t'',\boldsymbol{Y}'',\boldsymbol{x}'')$ into the list $\mathcal{L}$ so that the values of the third field of the records in $\mathcal{L}$ increase;

if $\omega > \mu$, then set $\omega := \mu$ and clean up the list $\mathcal{L}$, i.e. remove from it all such records $(\boldsymbol{Q},\boldsymbol{r},\upsilon,W,s,t,\boldsymbol{Y},\boldsymbol{x})$, that $\upsilon > \omega$;

`END DO`

to be tuned and determined under specific circumstances. We can, therefore, speak about a whole *class of methods* based on the common general idea of partitioning the interval parameters of the system.

To start the algorithm of Table 7, which we shall call `LinPPS1`, one should

- Find crude enclosures for the solution set and the inverse interval matrix, that is, compute $\boldsymbol{x} \supseteq \Xi(\boldsymbol{A}, \boldsymbol{b})$ and $\boldsymbol{Y} \supseteq \boldsymbol{A}^{-1}$,
- Assign the accuracy $\epsilon > 0$,
- Set $\Upsilon(\boldsymbol{A}, \boldsymbol{b}) := \underline{\boldsymbol{x}}$ and $\omega := +\infty$,
- Initialize working the list $\mathcal{L}$ by the record $(\boldsymbol{A}, \boldsymbol{b}, \underline{\boldsymbol{x}}, \boldsymbol{Y}, \boldsymbol{x})$.

To start the algorithm of Table 8, called `LinPPS2`, one needs accomplish the first three items as in the previous case, then set $W := 0$, $s := 0$, $t := 0$ for the check matrix W and check vectors s and t introduced in §3.3 and initialize the working list $\mathcal{L}$ by the record $(\boldsymbol{A}, \boldsymbol{b}, \underline{\boldsymbol{x}}, W, s, t, \boldsymbol{Y}, \boldsymbol{x})$.

Parallelization is another important point. Similar to its nearest relatives — "branch-and-bound" based interval global optimization techniques from [3,4,12] — our parameter partition methods for interval systems of equations are readily adapted for parallel processing, but deeper inquiring into this issue is beyond the scope of the present work.

References

1. Shary, S.P.: Reliab. Comput. 8, 321–419 (2002)
2. Alefeld, G., Herzberger, J.: Introduction to interval computations. Academic Press, New York (1983)
3. Hansen, E., Walster, G.W.: Global optimization using interval analysis. Marcel Dekker, New York (2004)
4. Kearfott, R.B.: Rigorous global search: continuous problems. Kluwer, Dordrecht (1996)
5. Neumaier, A.: Interval methods for systems of equations. Cambridge University Press, Cambridge (1990)
6. Rohn, J.: Lin Algebra Appl. 126, 39–78 (1989)
7. Shary, S.P.: SIAM J. Numer. Anal. 32, 610–630 (1995)
8. Kreinovich, V., Lakeyev, A.V., Rohn, J., Kahl, P.: Computational complexity and feasibility of data processing and interval computations. Kluwer, Dordrecht (1997)
9. Kearfott, R.B., Nakao, M.T., Neumaier, A., Rump, S.M., Shary, S.P., van Hentenryck, P.: Standardized notation in interval analysis (2002), `http://www.nsc.ru/interval/INotation.pdf`
10. Beeck, H.: Computing 10, 231–244 (1972) (in German)
11. Nickel, K.: Computing 18, 15–36 (1977) (in German)
12. Ratschek, H., Rokne, J.: New computer methods for global optimization. Ellis Horwood–Halsted Press, Chichester–New York (1988)
13. Horst, R., Tuy, H.: Global optimization. deterministic approaches. Springer, Berlin (1995)
14. Shary, S.P.: Interval Comput. 2(4), 18–29 (1992)
15. Hansen, E.: On linear algebraic equations with interval coefficients. In: Hansen, E. (ed.) Topics in interval analysis, pp. 35–46. Clarendon Press, Oxford (1969)