65 (2020) APPLICATIONS OF MATHEMATICS No. 5, 645–663

COMPLEXITY OF COMPUTING INTERVAL MATRIX POWERS FOR SPECIAL CLASSES OF MATRICES

David Hartman, Milan Hladík, Praha

Received December 31, 2019. Published online September 7, 2020.

Abstract. Computing powers of interval matrices is a computationally hard problem. Indeed, it is NP-hard even when the exponent is 3 and the matrices only have interval components in one row and one column. Motivated by this result, we consider special types of interval matrices where the interval components occupy specific positions. We show that computing the third power of matrices with only one column occupied by interval components can be solved in cubic time; so the asymptotic time complexity is the same as for the real case (considering the textbook matrix product method). We further show that for a fixed exponent k and for each interval matrix (of an arbitrary size) whose kth power has components that can be expressed as polynomials in a fixed number of interval variables, the computation of the kth power is polynomial up to a given accuracy. Polynomiality is shown by using the Tarski method of quantifier elimination. This result is used to show the polynomiality of computing the cube of interval band matrices, among others. Additionally, we study parametric matrices and prove NP-hardness already for their squares. We also describe one specific class of interval parametric matrices that can be squared by a polynomial algorithm.

Keywords: matrix power; interval matrix; interval computations; NP-hardness

MSC 2020: 65G40, 15Bxx

1. Introduction

Many real-world problems with origin in the system control theory are described by systems of linear differential equations, whose solutions are characterized by matrix powers in the discrete time case and by matrix exponentials in the continuous time case. Interval analysis is a tool that helps to handle uncertainty in the description of their state-space caused by inexactness of measurements [1], for example when using interval impulse response to efficiently design a robust controller [26]. The key

The research has been supported by the Czech Science Foundation Grant P403-18-04735S.

DOI: 10.21136/AM.2020.0379-19

problem when determining the impulse response for an uncertain system is to find the power of the corresponding interval matrix. Related problems dealing with the computation of the matrix exponential were studied in [8], [20].

Improving the complexity of computing an interval matrix power is thus a well motivated task. The problem of computing the square can be solved in polynomial time [15] by employing interval arithmetic. However, due to the properties of interval arithmetic, we cannot utilize it to compute the higher powers exactly. Indeed, it was shown that already computing the cube of an interval matrix is an NP-hard problem. For this reason, the problem of computing interval matrix powers was addressed using approximation algorithms [12]. Mayer [18] also studied interval matrix powers motivated by studies of initial value problem stability.

Recently, for a specific class of interval matrices, the problem has been shown to be polynomially tractable [13]—namely for a class of diagonally interval matrices, i.e., those having intervals on the diagonal while leaving remaining components to be real-valued. Motivated by this recent result, we explore the problem for other classes of interval matrices.

1.1. Computing interval matrix powers. First, we introduce some notions necessary for handling intervals and interval matrices. By convention, an interval means a compact real interval. A (*square*) *interval matrix* is defined as

$$\boldsymbol{A} = [\underline{A}, \bar{A}] = \{A \in \mathbb{R}^{n\times n};\ \underline{A} \leqslant A \leqslant \bar{A}\},$$

where $\underline{A} \leqslant \bar{A}$ are given matrices and inequalities are considered elementwise. To avoid any confusion, we refer to the particular interval items $\boldsymbol{A}_{ij}$ (or $\boldsymbol{a}_{ij}$) of an interval matrix $\boldsymbol{A}$ as *components*, and we refer to real matrices $A \in \boldsymbol{A}$ as *instances*. We denote the set of all n-by-n interval matrices by $\mathbb{IR}^{n\times n}$. By

$$A^c := \frac{1}{2}(\underline{A} + \bar{A}), \quad A^\Delta := \frac{1}{2}(\underline{A} - \bar{A})$$

we denote the midpoint and the radius of $\boldsymbol{A}$, respectively. When referring to the components of an interval matrix $\boldsymbol{A}$, we use A^c_{ij} and A^Δ_{ij} for the midpoint and the radius of $\boldsymbol{A}_{ij}$, respectively. When $\underline{A}_{i,j} = \bar{A}_{i,j}$, we call the (i,j)th component of $\boldsymbol{A}$ *degenerate*. Note that for a degenerate component $\boldsymbol{A}_{ij}$ we have $A^\Delta_{ij} = 0$.

An *enclosure* of a bounded set $\mathcal{B} \subset \mathbb{R}^{n\times n}$ is any $\boldsymbol{B} \in \mathbb{IR}^{n\times n}$ such that $\mathcal{B} \subseteq \boldsymbol{B}$. The *interval hull* of $\mathcal{B}$ is denoted by $\square\mathcal{B}$ and it is the smallest enclosure of $\mathcal{B}$, that is, $\square\mathcal{B} := \bigcap_{\mathcal{B}\subseteq\boldsymbol{B}\in\mathbb{IR}^{n\times n}} \boldsymbol{B}$.

We define the *kth interval matrix power* or *kth power of interval matrix* $\boldsymbol{A}$ as

$$\boldsymbol{A}^k := \{A^k;\ A \in \boldsymbol{A}\}.$$

Since this is not an interval matrix in general, we are content with its interval hull

$$[\boldsymbol{A}^k] := \square\{A^k;\ A \in \boldsymbol{A}\}$$

instead. Not all components of $\boldsymbol{A}$ have to be non-degenerate intervals. Let m denote the number of non-degenerate interval components, that is,

$$m := |\{(i,j);\ A_{ij}^{\Delta} > 0\}|.$$

Since the problem of computing matrix powers depends on an efficient realization of sequences of the corresponding arithmetic operations, it is reasonable to introduce basic operations with intervals [19]. Let $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{IR}$ and let $\circ$ be an arbitrary real binary operation. Then $\boldsymbol{a} \circ \boldsymbol{b} = \{a \circ b;\ a \in \boldsymbol{a}, b \in \boldsymbol{b}\}$. In particular, for arithmetic operations we can write simpler expressions

$$\begin{aligned}
\boldsymbol{a} + \boldsymbol{b} &= [\underline{a} + \underline{b}, \bar{a} + \bar{b}], \\
\boldsymbol{a} - \boldsymbol{b} &= [\underline{a} - \bar{b}, \bar{a} - \underline{b}], \\
\boldsymbol{a} \cdot \boldsymbol{b} &= [\min(M), \max(M)], \text{ where } M := \{\underline{a}\underline{b}, \underline{a}\bar{b}, \bar{a}\underline{b}, \bar{a}\bar{b}\}.
\end{aligned}$$

We can further generalize this approach from simple arithmetic operations to more general functions. For a given function $f = f(x_1, x_2, \ldots, x_n)$, the *image* of a set of interval variables $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n$ is

$$f(\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n) = \{f(x_1, x_2, \ldots, x_n);\ x_1 \in \boldsymbol{x}_1, x_2 \in \boldsymbol{x}_2, \ldots, x_n \in \boldsymbol{x}_n\}.$$

For some basic functions, computing the image is easy. For instance, for the square function $f(x) = x^2$ and $\boldsymbol{x} \in \mathbb{R}$, we have

$$f(\boldsymbol{x}) = \begin{cases} [\min(\underline{x}^2, \overline{x}^2), \max(\underline{x}^2, \overline{x}^2)] & \text{if } 0 \notin \boldsymbol{x}, \\ [0, \max(\underline{x}^2, \overline{x}^2)] & \text{otherwise,} \end{cases}$$

which we will utilize later in the paper. In general, however, computing the image is an NP-hard problem [16]. Note also that for a given function f we can have several equivalent expressions, each one of which can be used to compute an enclosure of the image using interval arithmetic; computing a matrix power is a good example. Moreover, some of these enclosures are possibly tight; see examples in [19]. If we consider the components of the original matrix as variables, we can express any component of the matrix power as a polynomial in these variables. Moreover, if some of the original components are degenerate intervals, we can simply use them as real coefficients in this polynomial.

We can obtain the exact image by interval arithmetic if an expression of the function meets the condition called single-use of expression (SUE). An expression satisfies SUE if every variable x_i appears only once there [9]. Considering this condition, some of the functions can be presented in an equivalent form that enables exact evaluation—this idea was used to prove that computing the square of an interval matrix is polynomially solvable [16].

1.2. Solving the hard case. NP-hardness of computing the cube of an interval matrix was shown [15] using NP-hardness of computing the matrix norm [22]

$$\|A\|_{\infty,1} = \max_{\|x\|_\infty=1} \|Ax\|_1.$$

This norm has many applications including the proof of intractability of computation of the regularity radius [21], which is defined as the distance from a given real matrix to the nearest singular one in maximum norm. Computing the regularity radius can be handled via approximation algorithms [10], or by solving this problem for specific classes of matrices [11]. In this paper we also focus on special classes of matrices to study the borderline between polynomiality and NP-hardness of the problem in question.

NP-completeness of the interval matrix power problem can be shown via a construction of a specific interval matrix for which the corresponding third power leads to the computation of the above-mentioned norm [15]. This interval matrix $\boldsymbol{A}$ can be defined as follows. Let B be any real-valued matrix. Then

$$\mathbf{A} = \begin{pmatrix} 0 & \mathbf{U} \\ L & 0 \end{pmatrix}, \tag{1.1}$$

where

$$L = \left(\begin{array}{c|ccc} 0 & \dots & 0 & \dots \\ \hline \vdots & & & \\ 0 & & B & \\ \vdots & & & \end{array}\right), \quad \mathbf{U} = \left(\begin{array}{c|ccc} 0 & \mathbf{x}_1 & \dots & \mathbf{x}_n \\ \hline \mathbf{y}_1 & & & \\ \vdots & & 0 & \\ \mathbf{y}_n & & & \end{array}\right).$$

Using a proper rearrangement, we can reduce the computation of the third power of this matrix to the computation of the above-mentioned norm [15]. Note that the interval components in this matrix are located only in the submatrix $\mathbf{U}$ containing one row and one column with non-degenerate intervals. This suggests that interval matrices having at least one row and one column composed of $\mathcal{O}(n)$ non-degenerate intervals lead to NP-completeness. On the other hand, the second author [13] shows an example of an interval matrix with a non-trivial structure for which the computation can be reduced to a polynomial one—namely an interval matrix that has only

the diagonal components non-degenerate. Motivated by both these results, we aim to find the most "complex" interval matrix for which the computation of the third power can be done in polynomial time.

Let us mention another important phenomenon. When showing that the computation of the interval matrix power is a polynomial problem for a particular class, we can always do it only up to a given accuracy: It cannot be computed exactly in rational arithmetic, since the values might be irrational as Example 1.1 shows (notice they are still algebraic numbers).

E x a m p l e 1.1. Consider the interval matrix

$$\boldsymbol{A} = \begin{pmatrix} [1,2] & 3 \\ -1 & 0 \end{pmatrix}.$$

Then $(A^3)_{1,1} = a_{1,1}^3 - 6a_{1,1}$ and its smallest value on the interval $a_{1,1} \in \boldsymbol{a}_{1,1} = [1,2]$ is the irrational value of $-4\sqrt{2}$, which is attained at $a_{1,1} = \sqrt{2}$. The third power of the whole matrix is

$$[\boldsymbol{A}^3] = \begin{pmatrix} [-4\sqrt{2}, -4] & [-6, 3] \\ [-1, 2] & [-6, -3] \end{pmatrix}.$$

2. Matrices with linearly many interval components

In the introduction we mention two different interval matrices having $\mathcal{O}(n)$ non-degenerate interval components for which different complexity results are known. For matrices having non-degenerate intervals only on the diagonal, there is a polynomial algorithm to compute their powers, while powers of matrices given in equation (1.1) cannot be handled polynomially. For this reason, a natural choice for further testing is represented by interval matrices having non-degenerate intervals only in one of their columns while leaving the remaining components real.

2.1. Third power of matrices with one interval column. Consider a class of interval matrices, where each matrix has one column occupied by non-degenerate interval components, while the remaining are degenerate. Note that this is equivalent to considering a row instead of a column. Moreover, without loss of generality, we can assume that such a column is the last one, i.e., the matrix can be written as follows

$$\boldsymbol{A} = \begin{pmatrix} a_{1,1} & a_{1,2} & \dots & a_{1,n-1} & \boldsymbol{a}_{1,n} \\ a_{2,1} & a_{2,2} & & & \boldsymbol{a}_{2,n} \\ \vdots & & \ddots & & \vdots \\ & & & a_{n-1,n-1} & \boldsymbol{a}_{n-1,n} \\ a_{n,1} & \dots & & a_{n,n-1} & \boldsymbol{a}_{n,n} \end{pmatrix}. \tag{2.1}$$

To analyse the complexity of computing the third power of this interval matrix we can write down polynomials for each component of the matrix power and try to express them in forms that are convenient for evaluation. This simplification aims either to obtain a form for which the SUE condition is met or to obtain a form for which we are able to evaluate the corresponding polynomial using a different approach. As a first step, we evaluate the components of the square of a real matrix.

Proposition 2.1. *Let $A \in \mathbb{R}^{n\times n}$. Then the components of the matrix $B := A^2$ can be written as*

$$B_{i,j} = \sum_{k=1}^{n-1} a_{i,k}a_{k,j} + a_{i,n}a_{n,j} \quad \forall\, i \in \{1,\dots,n\},\ j \in \{1,\dots,n-1\}, \tag{2.2}$$

$$B_{i,n} = \sum_{\substack{k=1\\k\neq i}}^{n-1} a_{i,k}a_{k,n} + a_{i,n}(a_{n,n} + a_{i,i}) \quad \forall\, i \in \{1,\dots,n-1\}, \tag{2.3}$$

and

$$B_{n,n} = \sum_{k=1}^{n-1} a_{n,k}a_{k,n} + a_{n,n}^2. \tag{2.4}$$

P r o o f. All equations are obtained via a simple application of matrix multiplication. Additionally, we adjust the expressions to avoid multiple occurrences of interval parameters $a_{1,n},\dots,a_{n,n}$. □

Using the expressions from Proposition 2.1, the components of the interval matrix $\boldsymbol{B} := [\boldsymbol{A}^2]$ can be computed by interval arithmetic combined with basic interval functions (herein, the square of an interval). This is due to the fact that the SUE condition is fulfilled.

▷ $\boldsymbol{B}_{i,j}$, $\forall\, i,j\colon\ j \neq n$: all components except the components from the last column

$$\boldsymbol{B}_{i,j} = \sum_{k=1}^{n-1} a_{i,k}a_{k,j} + \boldsymbol{a}_{i,n}a_{n,j}$$

▷ $\boldsymbol{B}_{i,n}$, $\forall\, i \neq n$: last column components except the last one

$$\boldsymbol{B}_{i,n} = \sum_{\substack{k=1\\k\neq i}}^{n-1} a_{i,k}\boldsymbol{a}_{k,n} + \boldsymbol{a}_{i,n}(\boldsymbol{a}_{n,n} + a_{i,i})$$

▷ $\boldsymbol{B}_{n,n}^2$: the last bottom-right component

$$\boldsymbol{B}_{n,n} = \sum_{k=1}^{n-1} a_{n,k}\boldsymbol{a}_{k,n} + \boldsymbol{a}_{n,n}^2$$

Since we are interested in computing the cube of an interval matrix, we now derive expressions for the components of the cube of ordinary matrices.

Lemma 2.2. *Let $A \in \mathbb{R}^{n\times n}$. Then the components of the matrix $C := A^3$ can be written as*

$$C_{i,j} = \sum_{l=1}^{n-1}\sum_{k=1}^{n-1} a_{i,k}a_{k,l}a_{l,j} + a_{i,n}\Big(\sum_{l=1}^{n-1} a_{n,l}a_{l,j} + a_{n,j}(a_{n,n}+a_{i,i})\Big) + a_{n,j}\sum_{\substack{k=1\\k\neq i}}^{n-1} a_{i,k}a_{k,n} \quad \forall\, i,j \in \{1,2,\dots,n-1\}, \tag{2.5}$$

$$C_{n,j} = \sum_{l=1}^{n-1}\sum_{k=1}^{n-1} a_{n,k}a_{k,l}a_{l,j} + a_{n,n}\sum_{l=1}^{n-1} a_{n,l}a_{l,j} + a_{n,j}\sum_{k=1}^{n-1} a_{n,k}a_{k,n} + a_{n,n}^2 a_{n,j} \quad \forall\, j \in \{1,2,\dots,n-1\}, \tag{2.6}$$

$$C_{i,n} = \sum_{\substack{l=1\\l\neq i}}^{n-1} a_{l,n}\Big(\sum_{k=1}^{n-1} a_{i,k}a_{k,l} + a_{i,n}a_{n,l} + a_{n,n}a_{i,l}\Big) + a_{i,n}\sum_{k=1}^{n-1} a_{i,k}a_{k,i} + a_{i,n}^2 a_{n,i} + a_{i,n}a_{n,n}^2 + a_{i,n}a_{n,n}a_{i,i} \quad \forall\, i \in \{1,2,\dots,n-1\}, \tag{2.7}$$

$$C_{n,n} = \sum_{l=1}^{n-1} a_{l,n}\Big(\sum_{k=1}^{n-1} a_{n,k}a_{k,l} + 2a_{n,n}a_{n,l}\Big) + a_{n,n}^3. \tag{2.8}$$

P r o o f. To compute the third power, we can start with the expressions for the square B derived in Proposition 2.1 and multiply them by the matrix A. To achieve the expression in (2.5), we need to use the expressions from equations (2.2) and (2.3). Let $i,j \in \{1,2,\dots,n-1\}$. Then

$$\begin{aligned} C_{i,j} &= \sum_{l=1}^{n-1} B_{i,l}a_{l,j} + B_{i,n}a_{n,j} \\ &= \sum_{l=1}^{n-1}\Big(\sum_{k=1}^{n-1} a_{i,k}a_{k,l} + a_{i,n}a_{n,l}\Big)a_{l,j} + \Big(\sum_{\substack{k=1\\k\neq i}}^{n-1} a_{i,k}a_{k,n} + a_{i,n}(a_{n,n}+a_{i,i})\Big)a_{n,j} \\ &= \sum_{l=1}^{n-1}\sum_{k=1}^{n-1} a_{i,k}a_{k,l}a_{l,j} + a_{i,n}\sum_{l=1}^{n-1} a_{n,l}a_{l,j} + a_{n,j}\sum_{\substack{k=1\\k\neq i}}^{n-1} a_{i,k}a_{k,n} + a_{i,n}a_{n,j}(a_{n,n}+a_{i,i}), \end{aligned}$$

from which (2.5) follows. For the components in the last row of C, see equation (2.6), we use the expressions from equations (2.3) and (2.4). For all $j \in \{1, 2, \ldots, n-1\}$ we have

$$\begin{aligned} C_{n,j} &= \sum_{l=1}^{n-1} B_{n,l} a_{l,j} + B_{n,n} a_{n,j} \\ &= \sum_{l=1}^{n-1} \Bigg(\sum_{k=1}^{n-1} a_{n,k} a_{k,l} + a_{n,n} a_{n,l} \Bigg) a_{l,j} + \Bigg(\sum_{k=1}^{n-1} a_{n,k} a_{k,n} + a_{n,n}^2 \Bigg) a_{n,j}, \end{aligned}$$

from which (2.6) follows. For all $i \in \{1, 2, \ldots, n-1\}$, the components defined in equation (2.7) can be derived as

$$\begin{aligned} C_{i,n} &= \sum_{l=1}^{n-1} B_{i,l} a_{l,n} + B_{i,n} a_{n,n} \\ &= \sum_{l=1}^{n-1} \Bigg(\sum_{k=1}^{n-1} a_{i,k} a_{k,l} + a_{i,n} a_{n,l} \Bigg) a_{l,n} + \Bigg(\sum_{\substack{k=1 \\ k \neq i}}^{n-1} a_{i,k} a_{k,n} + a_{i,n} (a_{n,n} + a_{i,i}) \Bigg) a_{n,n} \\ &= \sum_{l=1}^{n-1} a_{l,n} \sum_{k=1}^{n-1} a_{i,k} a_{k,l} + a_{i,n} \sum_{l=1}^{n-1} a_{n,l} a_{l,n} \\ &\quad + a_{n,n} \sum_{\substack{k=1 \\ k \neq i}}^{n-1} a_{i,k} a_{k,n} + a_{i,n} a_{n,n}^2 + a_{i,n} a_{n,n} a_{i,i}, \end{aligned}$$

from which (2.7) follows. Finally, the components in equation (2.8) can be derived as

$$\begin{aligned} C_{n,n} &= \sum_{l=1}^{n-1} B_{n,l} a_{l,n} + B_{n,n} a_{n,n} \\ &= \sum_{l=1}^{n-1} \Bigg(\sum_{k=1}^{n-1} a_{n,k} a_{k,l} + a_{n,n} a_{n,l} \Bigg) a_{l,n} + \Bigg(\sum_{k=1}^{n-1} a_{n,k} a_{k,n} + a_{n,n}^2 \Bigg) a_{n,n} \\ &= \sum_{l=1}^{n-1} a_{l,n} \sum_{k=1}^{n-1} a_{n,k} a_{k,l} + a_{n,n} \sum_{l=1}^{n-1} a_{n,l} a_{l,n} + a_{n,n} \sum_{k=1}^{n-1} a_{n,k} a_{k,n} + a_{n,n}^3, \end{aligned}$$

from which (2.8) follows. □

Note that the particular forms of these expressions were motivated by an effort to achieve the lowest number of occurrences of the last column components of the original matrix, i.e., those that represent the interval variables. We will use the formulae from the previous proposition to express the interval images or enclosures

of the components of the third power of an interval matrix, yielding Proposition 2.3 below. In the case of (2.9) we get the exact image by simple interval arithmetic evaluation thanks to the SUE condition. In the other cases, we obtain only interval enclosures in general.

Proposition 2.3. *Let* $\boldsymbol{A}$ *be from* (2.1) *and* $\boldsymbol{C} := [\boldsymbol{A}^3]$. *Then*

$$\begin{aligned}(2.9)\qquad \boldsymbol{C}_{i,j} = {} & \sum_{l=1}^{n-1}\sum_{k=1}^{n-1} a_{i,k}a_{k,l}a_{l,j} + \boldsymbol{a}_{i,n}\Bigg(\sum_{l=1}^{n-1} a_{n,l}a_{l,j} + a_{n,j}(\boldsymbol{a}_{n,n} + a_{i,i})\Bigg) \\ & + a_{n,j}\sum_{\substack{k=1\\k\neq i}}^{n-1} a_{i,k}\boldsymbol{a}_{k,n} \quad \forall\, i,j \in \{1,2,\dots,n-1\},\end{aligned}$$

$$\begin{aligned}(2.10)\qquad \boldsymbol{C}_{n,j} \subseteq {} & \sum_{l=1}^{n-1}\sum_{k=1}^{n-1} a_{n,k}a_{k,l}a_{l,j} + \boldsymbol{a}_{n,n}\sum_{l=1}^{n-1} a_{n,l}a_{l,j} \\ & + a_{n,j}\sum_{k=1}^{n-1} a_{n,k}\boldsymbol{a}_{k,n} + \boldsymbol{a}_{n,n}^2 a_{n,j} \quad \forall\, j \in \{1,2,\dots,n-1\},\end{aligned}$$

$$\begin{aligned}(2.11)\qquad \boldsymbol{C}_{i,n} \subseteq {} & \sum_{\substack{l=1\\l\neq i}}^{n-1} \boldsymbol{a}_{l,n}\Bigg(\sum_{k=1}^{n-1} a_{i,k}a_{k,l} + \boldsymbol{a}_{i,n}a_{n,l} + \boldsymbol{a}_{n,n}a_{i,l}\Bigg) + \boldsymbol{a}_{i,n}\sum_{k=1}^{n-1} a_{i,k}a_{k,i} \\ & + \boldsymbol{a}_{i,n}^2 a_{n,i} + \boldsymbol{a}_{i,n}\boldsymbol{a}_{n,n}^2 + \boldsymbol{a}_{i,n}\boldsymbol{a}_{n,n}a_{i,i} \quad \forall\, i \in \{1,2,\dots,n-1\},\end{aligned}$$

$$(2.12)\qquad \boldsymbol{C}_{n,n} \subseteq \sum_{l=1}^{n-1} \boldsymbol{a}_{l,n}\Bigg(\sum_{k=1}^{n-1} a_{n,k}a_{k,l} + 2\boldsymbol{a}_{n,n}a_{n,l}\Bigg) + \boldsymbol{a}_{n,n}^3.$$

In cases (2.10)–(2.12), interval arithmetic overestimates in general. However, we can determine the exact images by other means. We begin with the most complicated case (2.11).

Proposition 2.4. *Each of the components* $\boldsymbol{C}_{i,n}$, $i = 1,\dots,n-1$, *is computable in quadratic time.*

P r o o f. Consider the system of lines in $\mathbb{R}^2$ determined by $n-2$ equations in variables $a_{i,n}, a_{n,n}$:

$$a_{i,n}a_{n,l} + a_{n,n}a_{i,l} + \sum_{k=1}^{n-1} a_{i,k}a_{k,l} = 0, \quad l \neq i,n.$$

This system of lines constitutes a system of cells. The number of cells is quadratic in n [2], and an enumeration of them can be also performed in quadratic time [5].

Thus, we enumerate all the cells intersecting the rectangle $\boldsymbol{a}_{i,n} \times \boldsymbol{a}_{n,n}$. Each cell uniquely determines the signs of coefficients of the parameters $a_{l,n}$, $l \neq i, n$, so that we can set their values at $\underline{a}_{l,n}$ or $\bar{a}_{l,n}$ accordingly; let us denote them by $\tilde{a}_{l,n}$. As a consequence, the computation of $\boldsymbol{C}_{i,n}$ reduces to the computation of the ranges of polynomials of the type

$$\sum_{\substack{l=1\\ l\neq i}}^{n-1} \tilde{a}_{l,n}\left(\sum_{k=1}^{n-1} a_{i,k}a_{k,l} + \boldsymbol{a}_{i,n}a_{n,l} + \boldsymbol{a}_{n,n}a_{i,l}\right)$$
$$+ \boldsymbol{a}_{i,n}\sum_{k=1}^{n-1} a_{i,k}a_{k,i} + \boldsymbol{a}_{i,n}^2 a_{n,i} + \boldsymbol{a}_{i,n}\boldsymbol{a}_{n,n}^2 + \boldsymbol{a}_{i,n}\boldsymbol{a}_{n,n}a_{i,i}.$$

However, this is a polynomial of a fixed degree in two variables. Its coefficients are evaluated in quadratic time, and the polynomial itself then can be computed in constant time, e.g., by employing optimality conditions. □

Having this expression resolved, we can go through all remaining cases and show that computing the cube of an interval matrix with only one column of non-degenerate interval components can be done in polynomial time.

Theorem 2.5. *Let $\boldsymbol{A}$ be the matrix defined in* (2.1). *Then the interval matrix $\boldsymbol{C} := [\boldsymbol{A}^3]$ can be computed in $\mathcal{O}(n^3)$.*

P r o o f. (1) *Tractability of particular cases.* Cases (2.9) and (2.11) have been already discussed in Propositions 2.3 and 2.4.

Case (2.10) is also easy, since we enumerate the first and the third terms by interval arithmetic, and the remainder represents a quadratic function in one variable $a_{n,n}$ with domain $\boldsymbol{a}_{n,n}$, which can be resolved analytically.

Computing a tight enclosure for case (2.12) is also a polynomial problem. The component $C_{n,n}$ is a linear function with respect to the parameters $a_{l,n} \in \boldsymbol{a}_{l,n}$, $l \neq n$. Therefore, its maximum (and similarly its minimum) is achieved for $a_{l,n} \in \{\underline{a}_{l,n}, \bar{a}_{l,n}\}$. Which one of the values is the right one depends on the signs of their coefficients. Let c_l be the value of $a_{n,n}$, for which $\sum_{k=1}^{n-1} a_{n,k}a_{k,l} + 2a_{n,n}a_{n,l} = 0$ (if it exists). Then $\boldsymbol{a}_{n,n}$ is split according to these values to at most n subintervals. On each subinterval, all coefficients have constant signs, so that the values of $a_{l,n} \in \{\underline{a}_{l,n}, \bar{a}_{l,n}\}$, $l \neq n$, can be fixed. Then the expressions get reduced to a cubic polynomial in variable $a_{n,n}$, which is easily resolved.

(2) *Overall complexity.* The overall computational complexity is analysed by investigating the four cases; each of them takes cubic time in total (for the corresponding components together). Case (2.9): Each component $\boldsymbol{C}_{i,j}$ needs quadratic

time and there is a quadratic number of components. So the direct evaluation will be too expensive, and we have to evaluate them in another way. Define a matrix $G \in \mathbb{R}^{(n-1)\times(n-1)}$ as $G_{ij} = \sum_{l=1}^{n-1}\sum_{k=1}^{n-1} a_{i,k}a_{k,l}a_{l,j}$. This matrix can be evaluated in cubic time, since it is the cube of A after removing the last row and the last column. Now, we evaluate the component $\boldsymbol{C}_{i,j}$, $i,j \in \{1,2,\ldots,n-1\}$, as

$$\boldsymbol{C}_{i,j} = G_{i,j} + \boldsymbol{a}_{i,n}\left(\sum_{l=1}^{n-1} a_{n,l}a_{l,j} + a_{n,j}(\boldsymbol{a}_{n,n} + a_{i,i})\right) + a_{n,j}\sum_{\substack{k=1\\k\neq i}}^{n-1} a_{i,k}\boldsymbol{a}_{k,n}.$$

Case (2.10): For each component, we need quadratic time to compute the coefficients of the polynomial. The rest takes constant time. Case (2.11): By Proposition 2.4, each component needs quadratic time. Case (2.12): By the above reasoning, we have to inspect a linear number of instances. For each of them we evaluate the polynomial in quadratic time. □

2.2. Third power of an interval companion matrix. For some classes of interval matrices having only one column composed of non-degenerate intervals we can find a polynomial algorithm for computing their cube in a simpler way. For a monic polynomial $p(x) = c_0 + c_1x + \ldots + c_{n-1}x^{n-1} + x^n$, the corresponding companion matrix is

$$C = \begin{pmatrix} 0 & 0 & \ldots & 0 & -c_0 \\ 1 & 0 & \ldots & & -c_1 \\ 0 & 1 & \ldots & 0 & -c_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \ldots & 1 & -c_{n-1} \end{pmatrix}.$$

We will consider an interval version of this matrix called an interval companion matrix, in which all coefficients c_i are considered as interval coefficients $\boldsymbol{c}_i$. Considering c_i's as variables, we can show that the third power of the companion matrix has components which can be expressed with at least one occurrence of a variable only in the last three columns. In column $n-2$, there is a copy of the last column from the original matrix. In column $n-1$, the component on row position $j \neq n$ can be expressed as $-c_{j-2} - c_{j-1}c_{n-1}$ and the component on row position $j = n$ as $-c_{j-2} - c_{n-1}^2$. The most complex terms are in the last column. On row positions $j \neq 1, 2, n$, they are expressed as

$$c_{j-2} + c_jc_{n-2} - c_{n-1}(-c_{j-1} + c_jc_{n-1}). \tag{2.13}$$

It is a linear function in variables $c_{j-2}, c_{j-1}, c_j, c_{n-2}$ with appropriate interval domains, so the maximum and minimum are achieved at their lower and upper bounds.

For each combination of them (there are $2^4 = 16$ possibilities), the expression reduces to a quadratic function of c_{n-1}, which is easily resolved. The component in the last column with row index $j = n$ can be expressed as

$$-c_{j-2} + c_{j-1}c_{n-1} - c_{n-1}(-c_{j-1} + c_{n-1}^2).$$

This is handled in a similar way as above. Let us also note that cases when j is equal to 1 or 2 can be handled similarly. For $j = 2$ the expression is the same as in equation (2.13) only with the first term c_{j-2} missing. For $j = 1$ the corresponding expression is $c_{j-1}(c_{n-2} - c_{n-1}^2)$. Altogether it leads to the following result.

Proposition 2.6. *The problem of computing the third power of an interval companion matrix can be solved in* $\mathcal{O}(n^2)$.

3. Powers with constant expressions

The components of a matrix power can be expressed as polynomials in variables corresponding to the components of the original matrix. This potentially leads to violation of the SUE condition when computing the interval matrix power. The previous section shows that the third power can be efficiently computed when we restrict ourselves to a particular class. We define another class of interval matrices generically via conditioning on the expressions of components of their powers.

Definition 3.1. Let k and m_e be fixed. We say that a class of interval matrices has *constant expression of kth power* if any component of their kth powers can be expressed as a polynomial in at most m_e interval variables.

Notice that a class of interval matrices from the above definition is not restricted in the size of interval matrices, which are potentially unbounded. As an example, consider the class of all tridiagonal interval matrices of all sizes with k fixed and $m_e = (5 \cdot 3^{k-1} - 3)/2$. This class satisfies the property of constant expression of kth power.

Theorem 3.2. *Let k be fixed and let $\boldsymbol{A}$ be an interval matrix having rational endpoints with constant expression of kth power. Then the computation of the kth power of $\boldsymbol{A}$ up to any given accuracy ε is a polynomial problem with respect to input data and* $\log(1/\varepsilon)$.

P r o o f. Following the assumption, each component of the kth power of $\boldsymbol{A}$ can be represented as the interval image of a polynomial of at most m_e variables of degree

at most k. To find tight interval enclosure of any such component means to find tight lower and upper bounds of the corresponding polynomial, where m_e variables vary in their interval domains. We will focus on the lower bound of $[\boldsymbol{A}^k]$; the upper bound is dealt with analogously.

Consider an arbitrary component of the matrix A^k. By the assumption, it can be expressed as a polynomial $p(x_1, x_2, \ldots, x_{m_e})$, where $x_1, x_2, \ldots, x_{m_e}$ are interval parameters coming from interval domains $\{\boldsymbol{a}_1, \boldsymbol{a}_2, \ldots, \boldsymbol{a}_{m_e}\}$. Our aim is to compute the left endpoint $\underline{p}$ of its image $[\underline{p}, \overline{p}] = p(\boldsymbol{a}_1, \boldsymbol{a}_2, \ldots, \boldsymbol{a}_{m_e})$. Let $\boldsymbol{y} \ni \underline{p}$ be an initial enclosure of the minimum. It can be computed by evaluating the interval matrix power by interval arithmetic, for instance.

Now, the basic idea of the method is to apply a binary search on $\boldsymbol{y}$, and iteratively make it tighter up to a given accuracy. To this end, we need to check if $y \leqslant \underline{p}$ for a given value y, that is, if y is a lower bound on $\underline{p}$. Inequality $y \leqslant \underline{p}$ equivalently reads

$$
\begin{aligned}
(3.1) \qquad & (\forall\, x_1)(\forall\, x_2)\ldots(\forall\, x_{m_e}) \\
& \quad (x_1 - \underline{a}_1 \geqslant 0 \wedge x_1 - \bar{a}_1 \leqslant 0)\ldots(x_{m_e} - \underline{a}_{m_e} \geqslant 0 \wedge x_{m_e} - \bar{a}_{m_e} \leqslant 0) \\
& \quad \Rightarrow p(x_1, x_2, \ldots, x_{m_e}) - y \geqslant 0.
\end{aligned}
$$

To check for (3.1), we utilize the Tarski elimination method [25] to eliminate all universal quantifiers from this formula and thus determine whether this formula is true or not. Note that the values $\underline{a}_i, \bar{a}_i$ and y in (3.1) are constant.

Let us describe the approach in an algorithmic way.

(1) Compute an initial interval enclosure $\boldsymbol{y}$ of $\underline{p}$.
(2) Test whether the midpoint satisfies $y^c \leqslant \underline{p}$.
(3) If the answer is yes, set $\boldsymbol{y} = [y^c, \overline{y}]$.
(4) If the answer is no, set $\boldsymbol{y} = [\underline{y}, y^c]$.
(5) Goto 2 unless $y^\Delta < \varepsilon$.

It remains to show the polynomiality of this algorithm. In step 2, we call the variant of Tarski's method from Collins [4]. To analyse the complexity of the quantifier elimination we need to examine the formula (3.1). We already know that the number of variables m_e and the exponent k are fixed. In the formula there are $M = 2m_e + 1$ polynomials, namely m_e polynomials $x_i - \underline{a}_i$, m_e polynomials $x_i - \bar{a}_i$ and one polynomial $p(x_1, x_2, \ldots, x_{m_e}) - y$. Each of these polynomials gives rise to an atomic formula, so the number of formulas is $a = 2m_e + 1$. The corresponding atomic formulas are $x_i - \underline{a}_i \geqslant 0$, $x_i - \bar{a}_i \leqslant 0$ and $p(x_1, x_2, \ldots, x_{m_e}) - y \leqslant 0$. Since m_e is fixed, values M and a are fixed as well.

There is, however, another important characteristic of the formula that plays its role when evaluating the overall complexity. This characteristic is related to the max-

imum length of representation of any polynomial coefficient. The original algorithm of Collins [4] considers integer coefficients and their lengths. We have a rational matrix as an input, but we can simply transform the situation into integer data by an appropriate multiplication. Due to this reason, we can call d the maximum length of representation of any polynomial coefficient and consider it as polynomially bounded.

Having all these descriptors of the formula set up, we can evaluate the complexity of the quantifier elimination, following Collins [4], as $(2k)^{2^{2m_e+8}} M^{2^{m_e+6}} d^3 a$. By the assumption of the theorem, the only nonconstant variable is d, resulting in complexity $\mathcal{O}(d^3)$.

We use this elimination step in binary search for the initial interval with radius y^{Δ}. Since we are able to provide an initial estimate of the enclosure with polynomial size using interval arithmetic, the overall complexity can be written as $\mathcal{O}(d^3 \log(y^{\Delta}/\varepsilon))$. □

We will present two classes of interval matrices satisfying the assumption of Theorem 3.2. The first one are interval matrices with a constant number of interval parameters, and the second one are interval band matrices.

Corollary 3.3. *Let k and m be fixed. Then the computation of the kth power of an interval matrix with m non-degenerate interval components up to a given accuracy is a polynomial problem.*

Definition 3.4. A matrix $A \in \mathbb{R}^{n \times n}$ is called an *l-band matrix* (or simply a band matrix) if $a_{i,j} = 0$ for every i, j such that $|i - j| > k$.

Note that specific values of k provide well known classes of matrices such as a diagonal matrix ($k = 0$), a tridiagonal matrix ($k = 1$) or a pentadiagonal matrix ($k = 2$). Note also that band matrices are traditionally defined using two parameters determining the width of the band in super- and subdiagonal directions. For the purpose of this paper and simplicity of arguments we assume equality of both parameters and introduce just one, as given in Definition 3.4. Let us recall a well-known observation about band-matrices.

Proposition 3.5. *Let A, B be two l-band matrices both of size $n \times n$. Then the matrix $A \cdot B$ is a $2l$-band matrix.*

Considering this property, we can express the components of A^k as polynomials of the components of A. As summarized in the following observation, this results in a fixed number of variables appearing in each of the polynomials, leading consequently to the fulfillment of the condition from Definition 3.1. The value of m_e is derived based on a combinatorial calculation with a first-order recurrence equation.

Proposition 3.6. *Let k be fixed and let $\boldsymbol{A}$ be an interval l-band matrix. Then $\boldsymbol{A}$ has constant expression of its kth power with*

$$m_e = \frac{(4l+1)(2l+1)^{k-1} - 2l - 1}{2l}.$$

Using this observation and Theorem 3.2, we can write the following theorem describing the complexity of computing interval powers of interval band matrices.

Corollary 3.7. *Let k, l be fixed and let $\boldsymbol{A}$ be an interval l-band matrix. The problem of computing the kth power of $\boldsymbol{A}$ up to a given accuracy is a polynomial problem.*

4. Powers of parametric interval matrices

Many problems that are tractable for interval matrices become problematic for parametric ones even when considering a linear parametric case, see [6], [23], [24]. A linear parametric matrix reads

$$A(p) := \sum_{q=1}^{m} p_q A^{(q)}, \quad p_q \in \boldsymbol{p}_q, \tag{4.1}$$

where $\boldsymbol{p}_1, \ldots, \boldsymbol{p}_m \in \mathbb{IR}$. Note that this represents a generalization of an interval matrix. We can evaluate the expression $\boldsymbol{A} := \sum_{q=1}^{m} \boldsymbol{p}_q A^{(q)}$ using interval arithmetic, reducing thus the problem to a standard, non-parametric interval case, however, at the cost of overestimation. Therefore, it is better to exploit the special structure of parametric matrices. The interval hull of the kth power draws

$$[A(\boldsymbol{p})^k] := \square\{A(p)^k;\ p \in \boldsymbol{p}\}.$$

We want to handle the parametric case exactly without relaxation to ordinary interval matrices, which makes the problem of computing the kth power different. This can also be observed in the following result showing that the problem of computing the squares for the parametric case is difficult compared to the ordinary interval case.

Theorem 4.1. *Computation of $[A(\boldsymbol{p})^2]$ is an NP-hard problem.*

P r o o f. Consider a quadratic form $p^T Ap$ on an interval domain $p \in \boldsymbol{p} \in \mathbb{IR}^n$. It is known [3], [16] that computing the endpoints of the interval spanned by the range

$$\{p^T Ap;\ p \in \boldsymbol{p}\}$$

is an NP-hard problem. Define a parametric matrix of size $n+1$ as follows (the empty positions can possess any values)

$$A(p) = \begin{pmatrix} 0 & p_1 & \dots & p_n \\ \sum\limits_j a_{1j}p_j & & & \\ \vdots & & & \\ \sum\limits_j a_{nj}p_j & & & \end{pmatrix}.$$

Then

$$(A(p)^2)_{11} = \sum_i p_i \sum_j a_{ij}p_j = p^T Ap.$$

Therefore, determining the value of $[A(\boldsymbol{p})]_{11}$ is NP-hard. □

In the same spirit as handling the hard problem of computing the cube for interval matrices we can study the square of a parametric matrix by restriction to specific cases.

Proposition 4.2. *Set*

$$M_i := (A^{(1)}_{i,*} \mid \dots \mid A^{(m)}_{i,*}), \quad N_j := (A^{(1)}_{*,j} \mid \dots \mid A^{(m)}_{*,j}),$$

and let r be fixed. The computation of $[A(\boldsymbol{p})^2]$ can be performed in polynomial time on a class of problems with $\operatorname{rank}(M_i N_j) \leqslant r$ *for each $i, j = 1, \dots, n$.*

P r o o f. For any i, j and $p \in \boldsymbol{p}$, we have

$$(A(p)^2)_{i,j} = \sum_{l=1}^{n} \left(\sum_{q=1}^{m} A^{(q)}_{i,l} p_q \right) \left(\sum_{q'=1}^{m} A^{(q')}_{l,j} p_{q'} \right) = \sum_{l=1}^{n} (M_i p)_l (N_j p)_l = p^T (M_i N_j) p.$$

Finding the exact bounds for $[A(\boldsymbol{p})^2]_{i,j}$ is thus reduced to finding the minimum and maximum of the quadratic form $p^T(M_i N_j)p$ on the box $\boldsymbol{p}$. This can be done in polynomial time [14] since the rank of $M_i N_j$ is fixed due to the assumption. □

Notice that the condition $\operatorname{rank}(M_i N_j) \leqslant r$ for all i, j is achieved if $\operatorname{rank}(M_i) \leqslant r$ for each i, or if $\operatorname{rank}(N_j) \leqslant r$ for each j.

We can also restate Theorem 3.2 for the parametric case. Since the linear parametric matrix defined in (4.1) is not an ordinary interval matrix, we need to restate Definition 3.1 as follows.

Definition 4.3. Let k and m_p be fixed. We say that a class of linear parametric matrices has *constant expression of kth power* if any component of their kth power can be expressed as a polynomial in at most m_p parameters.

Using this definition, we can write the variant of Theorem 3.2 for linear parametric matrices. To show that this theorem holds we can follow the same steps as in the proof of Theorem 3.2.

Theorem 4.4. *Let k be fixed, $\boldsymbol{p}_1, \ldots, \boldsymbol{p}_m$ be interval parameters with rational endpoints and $A^{(1)}, A^{(2)}, \ldots, A^{(m)}$ be n-by-n rational matrices. Let $A(p)$ be a linear parametric matrix defined as in* (4.1) *with constant expression of kth power. Then the computation of the kth power of $A(p)$ up to any given accuracy ε is a polynomial problem with respect to input data and* $\log(1/\varepsilon)$.

5. Conclusion

In this work we deal with computing matrix powers of interval and parametric matrices. It is an NP-hard problem already for the third powers. For this reason, we investigate problems of computing powers of special types of interval matrices to approach to the borderline between polynomiality and NP-hardness of the problem in question. Theorem 2.5 shows that the problem of computing the third power of an interval matrix (2.1) with only one column occupied by non-degenerate interval components can be solved in $\mathcal{O}(n^3)$. Additionally, Proposition 2.6 shows that computing the third power of an interval companion matrix can be done in $\mathcal{O}(n^2)$. Considering the one-column case it would be interesting to find the borderline between complexities if such a borderline exists.

P r o b l e m 5.1. Let $\boldsymbol{A}$ be an interval matrix defined in (2.1). Determine whether there exists k for which a problem of computing the kth power is NP-complete.

Along similar lines, it would be interesting to investigate computational complexity of higher powers of the so-called diagonally interval matrices [13] or other special classes of interval matrices.

For higher powers of interval matrices we show that the problem is polynomial under certain assumptions given in Definition 3.1. This means that the exponent k is fixed and each component of A^k can be expressed as a polynomial in a fixed number of interval variables. The main ingredient of the proof is Tarski's quantifier elimination technique. This shows theoretical polynomiality of the computation of the interval matrix power up to a given accuracy. However, due to its intrinsic complexity, it might be difficult to implement this method in practice. As an alternative

approach, one can utilize Bernstein polynomials. They turned out to provide a tight approximation for the range of a polynomial on a hypercube or a simplex; see [17] and other papers in volume 17 of Reliable Computing [7].

Further, we consider linear parametric matrices, as defined in (4.1). For them we show that already the problem of computing the square of a linear parametric matrix is an NP-hard problem. On the other hand, we also present a class of parametric matrices with polynomial complexity. Finally, we provide a variant of the above-mentioned Tarski method for linear parametric matrices.

References

[1] *H.-S. Ahn, K. L. Moore, Y. Q. Chen*: Iterative Learning Control: Robustness and Monotonic Convergence for Interval Systems. Communications and Control Engineering. Springer, London, 2007. zbl MR doi

[2] *R. C. Buck*: Partition of space. Am. Math. Mon. *50* (1943), 541–544. zbl MR doi

[3] *M. Černý, M. Hladík*: The complexity of computation and approximation of the t-ratio over one-dimensional interval data. Comput. Stat. Data Anal. *80* (2014), 26–43. zbl MR doi

[4] *G. E. Collins*: Quantifier elimination for real closed fields by cylindrical algebraic decomposition. Quantifier Elimination and Cylindrical Algebraic Decomposition. Texts and Monographs in Symbolic Computation. Springer, Wien, 1998, pp. 85–121. zbl MR doi

[5] *H. Edelsbrunner, L. J. Guibas*: Topologically sweeping an arrangement. J. Comput. Syst. Sci. *38* (1989), 165–194. zbl MR doi

[6] *J. Garloff, E. D. Popova, A. P. Smith*: Solving linear systems with polynomial parameter dependency with application to the verified solution of problems in structural mechanics. Optimization, Simulation, and Control. Springer Optimization and Its Applications 76. Springer, New York, 2013, pp. 301–318. zbl MR doi

[7] *J. Garloff, A. P. Smith*: Preface (Special issue on the use of Bernstein polynomials in reliable computing: A centennial anniversary). Reliab. Comput. *17* (2012), i–vii. MR

[8] *A. Goldsztejn, A. Neumaier*: On the exponentiation of interval matrices. Reliab. Comput. *20* (2014), 53–72. MR

[9] *E. R. Hansen*: Sharpness in interval computations. Reliab. Comput. *3* (1997), 17–29. zbl MR doi

[10] *D. Hartman, M. Hladík*: Tight bounds on the radius of nonsingularity. Scientific Computing, Computer Arithmetic, and Validated Numerics. Lecture Notes in Computer Science 9553. Springer, Cham, 2016, pp. 109–115. zbl MR doi

[11] *D. Hartman, M. Hladík*: Regularity radius: Properties, approximation and a not a priori exponential algorithm. Electron. J. Linear Algebra *33* (2018), 122–136. zbl MR doi

[12] *D. Hartman, M. Hladík, D. Říha*: Computing the spectral decomposition of interval matrices and a study on interval matrix power. Available at `https://arxiv.org/abs/1912.05275`.

[13] *M. Hladík*: An overview of polynomially computable characteristics of special interval matrices. Beyond Traditional Probabilistic Data Processing Techniques: Interval, Fuzzy etc. Methods and Their Applications. Studies in Computational Intelligence 835. Springer, Cham, 2020, pp. 295–310. doi

[14] *M. Hladík, M. Černý, M. Rada*: A new polynomially solvable class of quadratic optimization problems with box constraints. Available at `https://arxiv.org/abs/1911.10877`.

[15] *O. Kosheleva, V. Kreinovich, G. Mayer, H. T. Nguyen*: Computing the cube of an interval matrix is NP-hard. SAC '05: Proceedings of the 2005 ACM Symposium on Applied Computing, Volume 2. ACM, New York, 2005, pp. 1449–1453. doi

[16] *V. Kreinovich, A. Lakeyev, J. Rohn, P. Kahl*: Computational Complexity and Feasibility of Data Processing and Interval Computations. Applied Optimization 10. Kluwer Academic Publishers, Dordrecht, 1998. zbl MR doi

[17] *R. Leroy*: Convergence under subdivision and complexity of polynomial minimization in the simplicial Bernstein basis. Reliab. Comput. *17* (2012), 11–21. MR

[18] *G. Mayer*: Interval Analysis and Automatic Result Verification. De Gruyter Studies in Mathematics 65. De Gruyter, Berlin, 2017. zbl MR doi

[19] *R. E. Moore, R. B. Kearfott, M. J. Cloud*: Introduction to Interval Analysis. SIAM, Philadelphia, 2009. zbl MR doi

[20] *E. P. Oppenheimer, A. N. Michel*: Application of interval analysis techniques to linear systems. II. The interval matrix exponential function. IEEE Trans. Circuits Syst. *35* (1988), 1230–1242. MR doi

[21] *S. Poljak, J. Rohn*: Checking robust nonsingularity is NP-hard. Math. Control Signals Syst. *6* (1993), 1–9. zbl MR doi

[22] *J. Rohn*: Computing the norm $\|A\|_{\infty,1}$ is NP-hard. Linear Multilinear Algebra *47* (2000), 195–204. zbl MR doi

[23] *I. Skalna*: Parametric Interval Algebraic Systems. Studies in Computational Intelligence 766. Springer, Cham, 2018. zbl MR doi

[24] *I. Skalna, M. Hladík*: Direct and iterative methods for interval parametric algebraic systems producing parametric solutions. Numer. Linear Algebra Appl. *26* (2019), Article ID e2229, 24 pages. zbl MR doi

[25] *A. Tarski*: A Decision Method for Elementary Algebra and Geometry. RAND Corporation, Santa Monica, California, 1948. zbl MR

[26] *Y. M. Zhang, R. Kovacevic*: Robust control of interval plants: A time domain method. IEE Proc., Control Theory Appl. *144* (1997), 347–353. zbl doi

Authors' addresses: *David Hartman*, Computer Science Institute of Charles University, Malostranské nám. 25, 118 00 Praha 1, Czech Republic, and Institute of Computer Science of the Czech Academy of Sciences, Pod Vodárenskou věží 2, 182 07 Praha 8, Czech Republic, e-mail: `hartman@iuuk.mff.cuni.cz`; *Milan Hladík*, Charles University, Faculty of Mathematics and Physics, Department of Applied Mathematics, Malostranské nám. 25, 118 00 Praha 1, Czech Republic, e-mail: `hladik@kam.mff.cuni.cz`.