Computing the Cube of an Interval Matrix Is NP-Hard

Olga Kosheleva,
Vladik Kreinovich
Pan-American Center
for Earth and
Environmental Studies
University of Texas at El Paso
El Paso, TX 79968, USA
vladik@cs.utep.edu

Günter Mayer
Fachbereich Mathematik
Universität Rostock
Universitätsplatz 1
D-18051 Rostock, Germany
guenter.mayer@mathematik.uni-rostock.de

Hung T. Nguyen
Dept. of Math. Sciences
New Mexico State University
Las Cruces, NM 88003, USA
hunguyen@nmsu.edu

ABSTRACT

In many practical applications, we are interested in computing the product of given matrices and/or a power of a given matrix. In some cases, the initial matrices are only known with interval uncertainty. It turns out that under this uncertainty, there is a principal difference between the product of two matrices and the product of three (or more) matrices:

- on the one hand, it is more or less known that the problems of computing the exact range for the product of two matrices – and for the square of a matrix – are computationally feasible;
- on the other hand, we prove that the problems of computing the exact ranges for the product of three matrices – and for the third power of a matrix – are NP-hard.

Categories and Subject Descriptors

F.2.1 [**Theory of Computation**]: Analysis of Algorithms and Problem Complexity—*Numerical Algorithms and Problems*; G.1.3 [**Mathematics of Computing**]: Numerical Analysis—*Numerical Linear Algebra: Error analysis*

General Terms

Theory

1. WHY INTERVAL MATRICES

In many real-life situations, we do not know the exact value of a physical quantity x, we only know the interval $\mathbf{x}$ of possible values of x. This happens, e.g., if our information about x comes from measurement, and the only information that we have about the possible error of the measuring instrument is that this error is guaranteed not to exceed a certain bound Δ. In this case, if the measurement result is $\widetilde{x}$, then from the fact that $|\widetilde{x}-x|\le\Delta$, where x is the (unknown) actual value of the measured quantity, we can conclude that x belongs to the interval $\mathbf{x}\stackrel{\rm def}{=}[\widetilde{x}-\Delta,\widetilde{x}+\Delta]$.

In some physical situations, quantities form a matrix

$$A=\begin{pmatrix} a_{11} & \ldots & a_{1j} & \ldots & a_{1n}\\ \ldots & \ldots & \ldots & \ldots & \ldots\\ a_{i1} & \ldots & a_{ij} & \ldots & a_{in}\\ \ldots & \ldots & \ldots & \ldots & \ldots\\ a_{n1} & \ldots & a_{nj} & \ldots & a_{nn}\end{pmatrix}. \qquad (1)$$

A typical example of when matrices emerge is when we describe the system's dynamics, i.e., how it transforms from a state $s(t)=(s_1(t),\ldots,s_n(t))$ at a given moment of time t to the state $s(t+1)=(s_1(t+1),\ldots,s_n(t+1))$ at the next moment of time $t+1$. In general, we have a dependence $s_i(t+1)=f_i(s_1(t),\ldots,s_n(t))$.

In many practical situations, e.g., in control, we have a stable state $s^{(0)}$ that does not change in time, and we are interested in small deviations from this stable state $s(t)\approx s^{(0)}$, i.e., the situations when $\Delta s_i(t)\stackrel{\rm def}{=}s_i(t)-s_i^{(0)}$ is small. In terms of Δs_i, the dynamic equation can be written as follows: $\Delta s_i(t+1)=F_i(\Delta s_1(t),\ldots,\Delta s_n(t))$. Since the values $\Delta s_i(t)$ are small, we can often safely ignore the quadratic and higher order terms in the dependence F_i, and assume that the function F_i is linear. Since $\Delta s_1(t)=\ldots=\Delta s_n(t)=0$ leads to $\Delta s_i(t+1)=0$, this linear dependence has no free term, i.e.:

$$\Delta s_i(t+1)=\sum_{j=1}^n a_{ij}\cdot\Delta s_j(t), \qquad (2)$$

for some coefficients a_{ij}. These coefficients form a matrix A, and in terms of this matrix, the dynamic equations take the form of a matrix product: $\Delta s(t+1)=A\,\Delta s(t)$.

In many real-life situations, we do not know the exact values of the quantities a_{ij}; for each i and j, we only know the interval $\mathbf{a}_{ij}$ of possible values of a_{ij}; see, e.g., [2, 3]. These intervals form an *interval matrix* in the following sense:

DEFINITION 1. *By an* square interval matrix *(or simply* interval matrix*, for short), we mean a square matrix whose elements are intervals:*

$$\mathbf{A}=\begin{pmatrix} \mathbf{a}_{11} & \ldots & \mathbf{a}_{1j} & \ldots & \mathbf{a}_{1n}\\ \ldots & \ldots & \ldots & \ldots & \ldots\\ \mathbf{a}_{i1} & \ldots & \mathbf{a}_{ij} & \ldots & \mathbf{a}_{in}\\ \ldots & \ldots & \ldots & \ldots & \ldots\\ \mathbf{a}_{n1} & \ldots & \mathbf{a}_{nj} & \ldots & \mathbf{a}_{nn}\end{pmatrix}. \qquad (3)$$

SAC'05 March 13-17, 2005, Santa Fe, New Mexico, USA

DEFINITION 2. *We say that a matrix A with entries a_{ij} is* consistent *with the information described by an interval matrix – and denote it by $A \in \mathbf{A}$ – if $a_{ij} \in \mathbf{a}_{ij}$ for all i and j.*

2. WHY PRODUCTS OF INTERVAL MATRICES

In many application problems, it is important to find the product of two or more matrices. For example, if the transition (2) from the moment t to the moment $t+1$ is described by a matrix A, and the transition from the moment $t+1$ to the moment $t+2$ is described by a matrix B, then the transition from the moment t to the moment $t+1$ can be described by the product matrix $C = BA$, with entries

$$c_{ij} = \sum_{k=1}^{n} a_{ik} \cdot b_{kj}. \quad (4)$$

So, in the situations when the only information that we have about A is that $A \in \mathbf{A}$, and the only information that we have about B is that $B \in \mathbf{B}$, we would like to know the resulting bounds on c_{ij}, i.e., we would like to know, for every i and j, the set (interval) of possible values:

DEFINITION 3. *By a* product *of two $n \times n$ interval matrices $\mathbf{A}$ and $\mathbf{B}$, we mean an interval matrix with the entries*

$$(\mathbf{AB})_{ij} \stackrel{\text{def}}{=} \{(AB)_{ij} \mid A \in \mathbf{A}, B \in \mathbf{B}\}. \quad (5)$$

Similar, for a transition from t to $t+3$, we are interested in the "product" of three interval matrices:

DEFINITION 4. *By a* product *of three $n \times n$ interval matrices $\mathbf{A}$, $\mathbf{B}$, and $\mathbf{C}$, we mean an interval matrix with the entries*

$$(\mathbf{ABC})_{ij} \stackrel{\text{def}}{=} \{(ABC)_{ij} \mid A \in \mathbf{A}, B \in \mathbf{B}, C \in \mathbf{C}\}. \quad (6)$$

How can we compute these products?

3. COMPUTING PRODUCTS OF INTERVAL MATRICES: TRADITIONAL APPROACH

The problem of computing the product of two or three matrices is a particular case of the following general problem: we have a function $f(x_1, \ldots, x_n)$ of n variables, we know the interval $\mathbf{x}_i$ of possible values of each of these variables, and we must find the range

$$f(\mathbf{x}_1, \ldots, \mathbf{x}_n) \stackrel{\text{def}}{=} \{f(x_1, \ldots, x_n) \mid x_1 \in \mathbf{x}_1, \ldots, x_n \in \mathbf{x}_n\} \quad (7)$$

of this function when $x_i \in \mathbf{x}_i$. This general problem is called the problem of *interval computations*; see, e.g., [5, 6, 13].

It is known that this range estimation problem is, in general, NP-hard [7]. Interval computations techniques enable us to either compute this range exactly, or at least to provide an enclosure for this range. For the case when $n = 2$ and the function $f(x_1, x_2)$ is one of the standard arithmetic operations ($+$, $-$, multiplication, etc.), there are known explicit formulas for the range of f. For example,

$$[\underline{x}_1, \overline{x}_1] + [\underline{x}_2, \overline{x}_2] = [\underline{x}_1 + \underline{x}_2, \overline{x}_1 + \overline{x}_2]; \quad (8)$$

$$[\underline{x}_1, \overline{x}_1] \cdot [\underline{x}_2, \overline{x}_2] = [\min(\underline{x}_1 \cdot \underline{x}_2, \underline{x}_1 \cdot \overline{x}_2, \overline{x}_1 \cdot \underline{x}_2, \overline{x}_1 \cdot \overline{x}_2),$$
$$\max(\underline{x}_1 \cdot \underline{x}_2, \underline{x}_1 \cdot \overline{x}_2, \overline{x}_1 \cdot \underline{x}_2, \overline{x}_1 \cdot \overline{x}_2)]. \quad (9)$$

These formulas form *interval arithmetic*; see, e.g., [5, 6, 13].

One way to compute the range for more complex functions f is to use straightforward ("naive") interval computations, i.e., replace each operation forming the algorithm f with the corresponding operation from interval arithmetic. This technique leads to an interval that is guaranteed to be an enclosure, i.e., to contain the desired range; it is known, however, that sometimes, this interval contains *excess width*, i.e., it is wider than the desired range [5, 6, 13].

An important case when straightforward interval computations lead to the exact range is the case of *single-use expressions* (SUE), when each variable x_i occurs only once; see, e.g., [4, 5]. So, we immediately arrive at the (known) feasible algorithm for computing the product of two interval matrices:

PROPOSITION 1. *There exists a feasible (polynomial-time) algorithm for computing the product of two interval matrices.*

PROOF. Indeed, the formula (4) is a SUE, so for the product $\mathbf{AB} = (\mathbf{c}_{ij})$ of two interval matrices, we can compute $\mathbf{c}_{ij}$ as follows:

$$\mathbf{c}_{ij} = \sum_{k=1}^{n} \mathbf{a}_{ik} \cdot \mathbf{b}_{kj}. \quad (10)$$

□

4. COMPUTING THE PRODUCT OF THREE INTERVAL MATRICES IS NP-HARD

For the product $D = ABC$ of three matrices, the expression $d_{ij} = \sum\limits_{k=1}^{n} \sum\limits_{l=1}^{n} a_{ik} \cdot b_{kl} \cdot c_{lj}$ is not SUE, so we can only guarantee that the straightforward interval computation leads to an enclosure. Actually, we can prove not only that it is not always exact, but that the problem of computing the exact product (6) of three interval matrices is NP-hard:

THEOREM 1. *The problem of computing the exact product of three square interval matrices is NP-hard.*

PROOF. Indeed, it is known [7, 14] that the following problem is NP-hard: given a square matrix $B = (b_{ij})$, compute the range of the sum $\sum\limits_{i=1}^{n} \sum\limits_{j=1}^{n} x_i \cdot b_{ij} \cdot y_j$ when $\mathbf{x}_i = \mathbf{y}_j = [-1, 1]$. This sum is a product of three matrices: $x^T B y$. To extend this result to $n \times n$ matrices A, B, and C, we can simply add 0s to x^T and y:

$$\mathbf{A} = \begin{pmatrix} \mathbf{x}_1 & \ldots & \mathbf{x}_n \\ 0 & \ldots & 0 \\ \ldots & \ldots & \ldots \\ 0 & \ldots & 0 \end{pmatrix}; \quad \mathbf{C} = \begin{pmatrix} \mathbf{y}_1 & 0 & \ldots & 0 \\ \mathbf{y}_2 & 0 & \ldots & 0 \\ \ldots & \ldots & \ldots & \ldots \\ \mathbf{y}_n & 0 & \ldots & 0 \end{pmatrix}.$$

For the product $D = ABC$, we have $d_{11} = x^T B y$. So, since computing the range of $x^T B y$ is NP-hard, computing the range $\mathbf{ABC}$ is also an NP-hard problem. □

5. WHY POWER OF A MATRIX

In many practical situations, we know that the system is stationary, i.e., that the transition from each moment of time to the next is described by the same matrix A. In this case, the transition from the moment t to the moment $t+2$ is described by the matrix A^2, the transition from the moment t to the moment $t+3$ is described by the matrix A^3, etc.

When we only know A with interval uncertainty, i.e., when we only know that $A \in \mathbf{A}$ for a given interval matrix A, then we would like to know, for every i and j, the set (interval) of possible values of A^2 and/or A^3:

DEFINITION 5. *By a* square $\mathbf{A}^2$ *of an interval matrix* $\mathbf{A}$, *we mean an interval matrix with entries*

$$(\mathbf{A}^2)_{ij} \stackrel{\text{def}}{=} \{(A^2)_{ij} \mid A \in \mathbf{A}\}.$$

DEFINITION 6. *By a* cube $\mathbf{A}^3$ *of an interval matrix* $\mathbf{A}$, *we mean an interval matrix with entries*

$$(\mathbf{A}^3)_{ij} \stackrel{\text{def}}{=} \{(A^3)_{ij} \mid A \in \mathbf{A}\}.$$

6. FEASIBLE ALGORITHM FOR COMPUTING THE SQUARE OF AN INTERVAL MATRIX

For $B = A^2$, the expression $b_{ij} = \sum\limits_{k=1}^{n} a_{ik} \cdot a_{kj}$ is not SUE. For example, for $i \neq j$, we have two occurrences of a_{ij}: $a_{ij} \cdot a_{jj}$ (when $k = j$) and $a_{ii} \cdot a_{ij}$ (when $k = j$). However, the problem is still feasible:

PROPOSITION 2. *There exists a feasible (polynomial-time) algorithm for computing the square of an interval matrix.*

PROOF. Indeed, the above occurrences are the only case when we have a multiple occurrence of a_{ij} in the formula for the square matrix b_{ij}. We can therefore reformulate the formulas for A^2 into the following equivalent SUE expression:

$$b_{ij} = \sum_{k: k \neq i, k \neq j} a_{ik} \cdot a_{kj} + a_{ij} \cdot (a_{ii} + a_{jj}) \quad (i \neq j). \tag{11}$$

Similarly, the expression for b_{ii} can be reformulated in the SUE form:

$$b_{ii} = \sum_{k: k \neq i} a_{ik} \cdot a_{ki} + a_{ii}^2. \tag{12}$$

By applying straightforward interval computations to these expressions, we get a feasible algorithm for computing the exact bounds for $\mathbf{A}^2$:

$$\mathbf{b}_{ij} = \sum_{k: k \neq i, k \neq j} \mathbf{a}_{ik} \cdot \mathbf{a}_{kj} + \mathbf{a}_{ij} \cdot (\mathbf{a}_{ii} + \mathbf{a}_{jj}) \quad (i \neq j); \tag{13}$$

$$\mathbf{b}_{ii} = \sum_{k: k \neq i} \mathbf{a}_{ik} \cdot \mathbf{a}_{ki} + \mathbf{a}_{ii}^2. \tag{14}$$

□

7. COMPUTING THE CUBE OF AN INTERVAL MATRIX: INTERVAL MATRIX PRODUCT IS NOT ASSOCIATIVE

For $\mathbf{A}^3$, we can, in principle, also use straightforward interval computations and compute the enclosure. It is worth mentioning that not only the resulting enclosure is not exact, but we get different results depending on the order in which we apply the matrix multiplication.

Namely, let us denote the straightforward product (10) of two interval matrices by $\mathbf{A} *_s \mathbf{B}$. Then, this operation is not even associative: in general, $(\mathbf{A} *_s \mathbf{A}) *_s \mathbf{A} \neq \mathbf{A} *_s (\mathbf{A} *_s \mathbf{A})$. Such an example was first given in [10, 11, 12]; in this paper, we give the simplest possible example of a 2×2 matrix $\mathbf{A}$ in which only one entry is known with interval uncertainty and all the values and endpoints of the intervals are equal to 0, 1, or -1:

$$\mathbf{A} = \begin{pmatrix} 1 & [0,1] \\ 1 & -1 \end{pmatrix}; \text{ then } \mathbf{A} *_s \mathbf{A} = \begin{pmatrix} [1,2] & [-1,1] \\ 0 & [1,2] \end{pmatrix};$$

$$\mathbf{A} *_s (\mathbf{A} *_s \mathbf{A}) = \begin{pmatrix} [1,2] & [-1,3] \\ [1,2] & [-3,0] \end{pmatrix};$$

$$(\mathbf{A} *_s \mathbf{A}) *_s \mathbf{A} = \begin{pmatrix} [0,3] & [-1,3] \\ [1,2] & [-2,-1] \end{pmatrix} \neq \mathbf{A} *_s (\mathbf{A} *_s \mathbf{A}).$$

$$\text{Here, } A = \begin{pmatrix} 1 & a_{12} \\ 1 & -1 \end{pmatrix}; \text{ so } A^2 = \begin{pmatrix} 1 + a_{12} & 0 \\ 0 & 1 + a_{12} \end{pmatrix};$$

$$A^3 = \begin{pmatrix} 1 + a_{12} & a_{12} + a_{12}^2 \\ 1 + a_{12} & -(1 + a_{12}) \end{pmatrix}; \text{ hence}$$

$$\mathbf{A}^2 = \begin{pmatrix} [1,2] & 0 \\ 0 & [1,2] \end{pmatrix}; \quad \mathbf{A}^3 = \begin{pmatrix} [1,2] & [0,2] \\ [1,2] & [-2,-1] \end{pmatrix}.$$

Comment. It should not be surprising that if we take uncertainty into consideration, then a previously associative operation becomes non-associative. For example, in *constructive mathematics*, a computable real number x is usually defined as an algorithm that maps a natural number k into a 2^{-k}-approximation $r(k)$ to the number x, i.e., into the rational number for which $|x - r(k)| \leq 2^{-k}$; see, e.g., [1] for the general introduction and [8, 9] for the software implementation.

If we know such algorithms r and s for two real numbers x and y, then we can construct the algorithm $r \oplus s$ for approximating the sum $x+y$: namely, $(r \oplus s)(k) = r(k+1) + s(k+1)$; from $|x - r(k+1)| \leq 2^{-(k+1)}$ and $|y - s(k+1)| \leq 2^{-(k+1)}$, we can conclude that

$$|(x + y) - (r(k+1) + s(k+1))| \leq$$

$$|x - r(k+1)| + |y - s(k+1)| \leq$$

$$2^{-(k+1)} + 2^{-(k+1)} = 2^{-k}.$$

If we have three numbers x, y, and z to add, with the algorithms r, s, and t, then we can construct the algorithm for approximating the sum $x + y + z$ as either $(r \oplus s) \oplus t$ or as $r \oplus (s \oplus t)$. Let us show that these two algorithms are, in general, different, i.e., that $\oplus$ is not associative.

PROPOSITION 3. *The operation* $\oplus$ *is not associative.*

PROOF. In general, $(r \oplus s)(k) = r(k+1) + s(k+1)$, so

$$((r \oplus s) \oplus t)(k) = (r \oplus s)(k+1) + t(k+1) =$$

$$r(k+2) + s(k+2) + t(k+1)$$

and, similarly,

$$r \oplus (s \oplus t)(k) = r(k+1) + s(k+2) + t(k+2).$$

As an example where these expressions differ, let us take three algorithms that approximate the real number 0: $r(k) = 2^{-k}$, $s(k) = 0$, and $t(k) = -2^{-k}$. Then, for every k, we have $((r \oplus s) \oplus t)(k) = 2^{-(k+2)} - 2^{-(k+1)} < 0$, while $(r \oplus (s \oplus t))(k) = 2^{-(k+1)} - 2^{-(k+2)} > 0$. So,

$$((r \oplus s) \oplus t)(k) \neq (r \oplus (s \oplus t))(k),$$

hence $(r \oplus s) \oplus t \neq r \oplus (s \oplus t)$. $\square$

Non-associativity also naturally emerges if we consider probabilistic uncertainty; see, e.g., [15].

8. COMPUTING THE CUBE OF AN INTERVAL MATRIX IS NP-HARD

Let us prove that, in general, computing $\mathbf{A}^3$ is NP-hard.

THEOREM 2. *The problem of computing the cube of an interval matrix is NP-hard.*

PROOF. For this proof, we will use the same result from [7, 14] as we used to prove that computing the product of three interval matrices is NP-hard: that, given a square matrix $B = (b_{ij})$, it is NP-hard to compute the range of the product $x^T By$, where $\mathbf{x}_i = \mathbf{y}_j = [-1, 1]$.

Specifically, for each $n \times n$ matrix B, we will consider the following $(2n+2) \times (2n+2)$ interval matrix:

$$\mathbf{A} = \begin{pmatrix} 0 & \mathbf{U} \\ L & 0 \end{pmatrix}, \tag{15}$$

where

$$L = \left(\begin{array}{c|c} 0 & \ldots \ 0 \ \ldots \\ \hline \ldots & \\ 0 & B \\ \ldots & \end{array}\right); \ \mathbf{U} = \left(\begin{array}{c|c} 0 & \mathbf{x}_1 \ \ldots \ \mathbf{x}_n \\ \hline \mathbf{y}_1 & \\ \ldots & 0 \\ \mathbf{y}_n & \end{array}\right)$$

Please note that L is a traditional (number-valued) matrix, i.e., a degenerate case of an interval matrix – that is why we denoted it by the font reserved for such matrices. On the other hand, the matrix $\mathbf{U}$ has non-degenerate interval entries and is, thus, a truly interval matrix.

For every matrix

$$A = \begin{pmatrix} 0 & U \\ L & 0 \end{pmatrix} \in \mathbf{A} = \begin{pmatrix} 0 & \mathbf{U} \\ L & 0 \end{pmatrix}, \tag{16}$$

we have

$$A^2 = \begin{pmatrix} 0 & U \\ L & 0 \end{pmatrix} \begin{pmatrix} 0 & U \\ L & 0 \end{pmatrix} = \begin{pmatrix} UL & 0 \\ 0 & LU \end{pmatrix}, \tag{17}$$

hence

$$A^3 = A^2 A = \begin{pmatrix} UL & 0 \\ 0 & LU \end{pmatrix} \begin{pmatrix} 0 & U \\ L & 0 \end{pmatrix} =$$

$$\begin{pmatrix} 0 & ULU \\ LUL & 0 \end{pmatrix}.$$

Here,

$$UL = \left(\begin{array}{c|c} 0 & x^T \\ \hline y & 0 \end{array}\right) \left(\begin{array}{c|c} 0 & 0^T \\ \hline 0 & B \end{array}\right) =$$

$$\left(\begin{array}{c|c} 0 & x^T B \\ \hline 0 & 0 \end{array}\right),$$

hence

$$ULU = \left(\begin{array}{c|c} 0 & x^T B \\ \hline 0 & 0 \end{array}\right) \left(\begin{array}{c|c} 0 & x^T \\ \hline y & 0 \end{array}\right) =$$

$$\left(\begin{array}{c|c} z & 0 \\ \hline 0 & 0 \end{array}\right),$$

where $z = x^T By$. So, $(ULU)_{11} = (A^3)_{1,n+2} = x^T By$. Since computing the range of $x^T By$ is NP-hard, computing the range $\mathbf{A}^3$ is also an NP-hard problem. $\square$

9. CONCLUSION

In many practical applications, the relation between the state $s(t) = (s_1(t), \ldots, s_n(t))$ at a moment of time t and the state $s(t+1)$ at the next moment of time is linear, i.e., has the form $s(t+1) = A(t)s(t)$ for some matrix $A(t)$ – that may be, in general, different for different moments of time t. In this situation, the transition over several time intervals, i.e., the transition from $s(t)$ to $s(t+k)$, where $k \geq 2$, is described by a formula $s(t+k) = Cs(t)$, where $C = A(t+k-1)A(t+k-2)\ldots A(t)$ is the product of k matrices $A(t+k-1)$, $\ldots$, $A(t)$. For stationary systems in which the transition matrix $A(t) = A$ is the same for all moments t, we get a similar formula $s(t+k) = Cs(k)$, where $C = A^k$. So, to describe such transitions, it is important to compute the product of given matrices and/or a power of a given matrix.

In some real-life cases, the initial matrices are only known with interval uncertainty. In such situations, different possible values $a_{ij}(t)$ of matrix entries for $A(t)$ lead, in general, to different entries c_{ij} in the resulting matrix C. It is desirable to find the exact range of values for these entries.

It is – more or less – known that the problems of computing the exact range for the product of two matrices – and for the square of a matrix – are computationally feasible. In this paper, we show that already for three matrices, the situation is radically different. Specifically, we prove that the problems of computing the exact ranges for the product of three matrices – and for the third power of a matrix – are NP-hard.

10. ACKNOWLEDGMENTS

This work was supported by by the German Research Council DFG, by NASA grant NCC5-209, by USAF grant F49620-00-1-0365, by NSF grants EAR-0112968, EAR-0225670, and EIA-0321328, by Army Research Laboratories grant DATM-05-02-C-0046, and by the NIH grant 3T34GM008048-20S1.

The authors are thankful to all the participants of the International Dagstuhl Seminar "Numerical Software with Result Verification" (Dagstuhl Castle, Germany, January 19–24, 2003) for valuable discussions, and to the anonymous referees for important comments.

11. REFERENCES

[1] O. Aberth, *Precise Numerical Analysis Using C++*, Academic Press, New York, 1998.

[2] B. R. Barmish, *New tools for robustness of linear systems*, McMillan, N.Y., 1994.

[3] S. P. Bhattacharyya, H. Chapellat, and L. Keel, *Robust Control: The Parametric Approach*, Prentice-Hall, Upper Saddle River, NJ, 1995.

[4] E. Hansen, Sharpness in interval computations, *Reliable Computing*, 1997, Vol. 3, pp. 7–29.

[5] L. Jaulin, M. Kieffer, O. Didrit, and E. Walter, *Applied Interval Analysis: With Examples in Parameter and State Estimation, Robust Control and Robotics*, Springer, London, 2001.

[6] R. B. Kearfott, *Rigorous Global Search: Continuous Problems*, Kluwer, Dordrecht, 1996.

[7] V. Kreinovich, A. Lakeyev, J. Rohn, and P. Kahl, *Computational complexity and feasibility of data processing and interval computations*, Kluwer, Dordrecht, 1997.

[8] D. Lester, Using PVS to validate the inverse trigonometric functions of an exact arithmetic, In: R. Alt, A. Frommer, R. B. Kearfott, and W. Luther (eds.), *Numerical Software with Result Verification*, (International Dagstuhl Seminar, Dagstuhl Castle, Germany, January 19–24, 2003), Springer Lectures Notes in Computer Science, 2004, Vol. 2991, pp. 274–305.

[9] D. Lester and P. Gowland, Using PVS to validate the algorithms of an exact arithmetic, *Theoretical Computer Science*, 2001, Vol. 291, pp. 203–218.

[10] G. Mayer, On the convergence of powers of interval matrices, *Linear Algebra and its Applications*, 1984, Vol. 58, pp. 201–216.

[11] G. Mayer, On the convergence of powers of interval matrices, Part II, *Numerische Mathematik*, 1985, Vol. 46, pp. 69–83.

[12] G. Mayer, Grundbegriffe der Intervallrechnung, In: U. Kulisch (ed.), *Wissenschaftlisches Rechnen mit Einführung*, Vieweg, Braunschweig, 1989, pp. 101–117.

[13] R. E. Moore, *Methods and Applications of Interval Analysis*, SIAM, Philadelphia, 1979.

[14] J. Rohn, Computing the Norm $\|A\|_1$ is NP-Hard, *Linear and Multilinear Algebra*, 2000, Vol. 47, pp. 195–204.

[15] R. Trejo, V. Kreinovich, I. R. Goodman, J. Martinez, and R. Gonzalez, A Realistic (Non-Associative) Logic And a Possible Explanations of 7 ± 2 Law, *International Journal of Approximate Reasoning*, 2002, Vol. 29, pp. 235–266.