submitted May 18, accepted November 28, 2011, published in Numerical Algorithms 61(1), 1-34 (2012).

FAST INTERVAL MATRIX MULTIPLICATION

SIEGFRIED M. RUMP *

Abstract. Several methods for the multiplication of point and/or interval matrices with interval result are discussed. Some are based on new priori estimates of the error of floating-point matrix products. The amount of overestimation including all rounding errors is analyzed. In particular, algorithms for conversion of infimum-supremum to midpoint-radius representation are discussed and analyzed, one of which is proved to be optimal. All methods are much faster than the classical method because almost no switch of the rounding mode is necessary, and because our methods are based on highly optimized BLAS3 routines. We discuss several possibilities to trade overestimation against computational effort. Numerical examples focussing in particular on applications using interval matrix multiplications are presented.

Key words. Interval arithmetic, overestimation, matrix multiplication, infimum-supremum representation, optimal midpoint-radius representation, interval matrix product, rounding mode, BLAS, unit in the first place (ufp), error analysis

AMS subject classifications. 15-04, 65G99, 65-04

1. Introduction. Interval arithmetic is a convenient though not mandatory way to obtain rigorous results on digital computers. Serious applications such as rigorous error bounds for the solution of partial differential equations [25, 24, 32, 33] or computer-assisted proofs [5] are based on an efficient implementation of interval arithmetic, where in particular interval matrix multiplication is a major time consuming part. Therefore we are interested in fast algorithms to that purpose as well as their analysis. Here "fast" refers to the practical execution time on today's computers, where obstacles like cache misses, branches, switching the rounding mode and compiler optimization require special attention.

Let $\mathbb{F}$ denote a set of binary floating-point numbers according to the IEEE 754 floating-point standard [12, 13]. Throughout the paper we assume that no overflow occurs, but allow underflow. Interval quantities with floating-point endpoints, i.e. the sets of scalar intervals, interval vectors and matrices are defined by

$$\mathbf{A} \in \mathbb{IF}^{m\times n} \;:\Leftrightarrow\; \mathbf{A} \in \{\, [A_1, A_2] \;:\; A_1, A_2 \in \mathbb{F}^{m\times n} \text{ and } A_1 \le A_2 \,\}\,, \tag{1.1}$$

where here and in the following interval matrices are written in bold letters, and comparison is to be understood componentwise. Such an interval quantity represents the set of all *real* matrices within the bounds, i.e.

$$[A_1, A_2] = \{\, A \in \mathbb{R}^{m\times n} \;: A_1 \le A \le A_2 \,\}\,.$$

Of course, the endpoints A_1, A_2 may be real matrices as well with similar definitions. Besides the infimum-supremum representation (1.1), we also use the midpoint-radius representation

$$\langle mA, rA\rangle := \{\, A \in \mathbb{R}^{m\times n} \;:\; mA - rA \le A \le mA + rA \} \,. \tag{1.2}$$

Any $A \in \mathbb{F}^{m\times n}$ can be identified with its "point interval" $\mathbf{A} := [A, A] = \langle A, 0\rangle$. The power set operations $\circ \in \{+, -, \cdot\}$ (in this paper we do not need division) between compatible interval quantities $\mathbf{A}, \mathbf{B}$ are defined by

$$\mathbb{PR}(\mathbf{A} \circ \mathbf{B}) := \{A \circ B : A \in \mathbf{A}, B \in \mathbf{B}\}\,. \tag{1.3}$$

*Institute for Reliable Computing, Hamburg University of Technology, Schwarzenbergstraße 95, Hamburg 21071, Germany, and Visiting Professor at Waseda University, Faculty of Science and Engineering, 3–4–1 Okubo, Shinjuku-ku, Tokyo 169–8555, Japan (rump@tu-harburg.de).

Interval operations between compatible interval quantities $\mathbf{A}, \mathbf{B}$ are defined by

$$\mathbf{A} \circ \mathbf{B} := \bigcap \{\mathbf{M} : \mathbb{PR}(\mathbf{A} \circ \mathbf{B}) \subseteq \mathbf{M}\} \tag{1.4}$$

with $\mathbf{M}$ denoting an interval quantity of appropriate dimension. The *inclusion principle* dictates that the interval result shall always be a superset of the power set result, and $\mathbf{A} \circ \mathbf{B}$ is the best possible inclusion with floating-point endpoints. The result of a scalar interval operation $\mathbf{c} = [c_1, c_2] = \mathbf{a} \circ \mathbf{b}$ can be computed directly using the bounds and floating-point arithmetic with directed rounding:

$$\begin{aligned} c_1 &:= \min\big(\ \mathrm{fl}_\nabla(a_1 \circ b_1)\ ,\ \mathrm{fl}_\nabla(a_1 \circ b_2)\ ,\ \mathrm{fl}_\nabla(a_2 \circ b_1)\ ,\ \mathrm{fl}_\nabla(a_2 \circ b_2)\ \big) \ \in\ \mathbb{F}\ , \\ c_2 &:= \max\big(\ \mathrm{fl}_\triangle(a_1 \circ b_1)\ ,\ \mathrm{fl}_\triangle(a_1 \circ b_2)\ ,\ \mathrm{fl}_\triangle(a_2 \circ b_1)\ ,\ \mathrm{fl}_\triangle(a_2 \circ b_2\ )\big) \ \in\ \mathbb{F}\ . \end{aligned}$$

This is no longer true for interval matrix multiplication. The classical way [28, Chapter 1] to compute an interval inclusion $\mathbf{C} \in \mathbb{IF}^{m \times n}$ of $\mathbb{PR}(\mathbf{AB})$ for $\mathbf{A} \in \mathbb{IF}^{m \times k}$, $\mathbf{B} \in \mathbb{IF}^{k \times m}$ is a 3-fold loop defining

$$\mathbf{C}_{ij} := \mathbf{A}_{i1} \cdot \mathbf{B}_{1j} + \ldots + \mathbf{A}_{ik} \cdot \mathbf{B}_{kj}\ , \tag{1.5}$$

where additions and multiplications are (scalar) interval operations. Classically, (1.5) is also used if one or both operands are point intervals. This is the basis for many libraries such as ACRITH [1], ARITHMOS [2], Intlib [16] or C-XSC [17]. We abbreviate it by $classical(\mathbf{A} \cdot \mathbf{B}) := \mathbf{C}$. Obviously $\mathbb{PR}(\mathbf{AB}) \subseteq \mathbf{A} \cdot \mathbf{B} \subseteq classical(\mathbf{A} \cdot \mathbf{B})$, and in general $\mathbb{PR}(\mathbf{AB}) \subsetneqq \mathbf{A} \cdot \mathbf{B} \subsetneqq classical(\mathbf{A} \cdot \mathbf{B})$.

The classical approach (1.5) is very slow on today's computers because i) it requires $2n^3$ times switching the rounding mode, thus ii) jeopardizing compiler optimization and iii) putting the burden of well-known acceleration techniques such as blocked code, parallel implementations etc. on the user.

A considerable improvement was achieved by Knüppel [18, 19] for point matrix times interval matrix, requiring only $2n^2$ switches of the rounding mode. In [36] midpoint-radius representation is used reducing the switches of rounding mode to 2, independent of the dimension. This is also true for the product of two interval matrices. This method, which is used in INTLAB [37], is particularly fast because it allows to use BLAS3 [3, 8] routines, sequential or parallel. BLAS3 routines such as $xGEMM$ for matrix multiplication may use blocking, multi-tasking and other ways to improve performance. A drawback of the midpoint-point radius approach is an inherent overestimation.

A timing comparison[1] for different dimensions of the classical approach (1.5), Knüppel's Profil [18, 19] and the midpoint-radius approach [36] is shown in Table 1.1. The gcc compiler version 4.4.1 on an Intel(R) Core(TM)2 Duo CPU with 3 Ghz was used, and the BLAS library by Kazushige Goto [8], a fast implementation which made it to the New York Times [21].

TABLE 1.1

Computing time for real times interval and interval times interval matrix multiplication (random data) in C using the classical approach (1.5), Knüppel's Profil [18, 19] and the midpoint-radius representation [36], one floating-point matrix multiplication using Goto-BLAS normed to 1.

	real × interval matrix			interval × interval matrix		
dimension	classical	Profil	mid-rad	classical	Profil	mid-rad
100	416	15	4.2	823	148	5.6
200	517	17	3.8	1025	184	5.5
500	646	21	3.7	1191	203	5.1
1000	716	30	3.4	1237	206	4.7
2000	774	36	3.3	1322	218	4.4

[1] Many thanks to Viktor Härter for performing the tests.

Historically the first paper using interval arithmetic to develop algorithms with result verification was Sunaga's master thesis [43]. It is hand-written in Japanese and received no attention. It covers many of the results in Moore's thesis [22], who undoubtedly popularized interval arithmetic. Midpoint-radius representation was used by Sunaga and subsequently also by other authors [6]. Most researchers including Moore, however, prefer infimum-supremum representation. The reason is the mentioned inherent overestimation due to a misplaced midpoint. In [36] it is shown that the overestimation for the product of two interval matrices is usually small, and it is universally limited by a factor 1.5. The latter, however, is only possible for very wide input data. This analysis neglects rounding errors. The overestimation can be avoided at the price of extra operations (cf. [28, Proposition 1.6.5], see also Subsection 4.3, in particular (4.19)ff.).

The substantial improvement shown in Table 1.1 is the motivation to analyze the overestimation (including rounding errors) of the classical and of the midpoint-radius approach compared to the best possible inclusion $\mathbf{A}\cdot\mathbf{B}$, to summarize existing and to present new and even faster algorithms for interval matrix multiplication. For the conversion of infimum-supremum to midpoint-radius representation three algorithms are presented and analyzed, and one of them is shown to be optimal.

The worst case factor 1.5 of overestimation seems discouraging, and this may be true when looking at the single matrix product. When using interval matrix multiplication in a practical application such as computing bounds for the solution of a linear system, things change. The second motivation is to show the influence of the overestimation in practical applications.

2. Notation and error estimates. The relative rounding error unit, the distance from 1.0 to the next smaller[2] floating-point number, is denoted by `eps`, and the underflow unit by `eta`, that is the smallest positive (subnormal) floating-point number. For IEEE 754 double precision (binary64) we have $\texttt{eps} = 2^{-53}$ and $\texttt{eta} = 2^{-1074}$. Denote by $\texttt{realmin} := \frac{1}{2}\texttt{eps}^{-1}\texttt{eta}$ the smallest positive normalized floating-point number, and by $\texttt{realmax} := \max\{f \in \mathbb{F}\}$ the largest representable floating-point number. Note that $\mathbb{F} \subseteq \texttt{eta}\mathbb{Z}$ and $-\texttt{realmax} \le f \le \texttt{realmax}$ for all finite $f \in \mathbb{F}$. Furthermore, the *underflow range* is defined by

$$\mathbb{U} := \{x \in \mathbb{R} : |x| < \texttt{realmin}\} \ . \tag{2.1}$$

The distance of a floating-point number $f \in 2\mathbb{U}$ to its neighbors is `eta`. For a very readable and thorough introduction to floating-point arithmetic cf. [23].

We denote by $\mathrm{fl}_\square : \mathbb{R} \to \mathbb{F}$ a rounding to nearest, that is

$$x \in \mathbb{R} \ : \ |\mathrm{fl}_\square(x) - x| = \min\{|f - x| : f \in \mathbb{F}\} \ . \tag{2.2}$$

Any rounding of the tie can be used without jeopardizing the following estimates, only (2.2) and $\mathrm{fl}_\square(-x) = -\mathrm{fl}_\square(x)$ must hold true. Moreover, we need rounding downwards $\mathrm{fl}_\nabla : \mathbb{R} \to \mathbb{F}$ and rounding upwards $\mathrm{fl}_\triangle : \mathbb{R} \to \mathbb{F}$ defined by

$$x \in \mathbb{R} \ : \quad \mathrm{fl}_\nabla(x) := \max\{f \in \mathbb{F} : f \le x\} \quad \text{and} \quad \mathrm{fl}_\triangle(x) := \min\{f \in \mathbb{F} : f \ge x\} \ . \tag{2.3}$$

In contrast to the rounding of a real number into a floating-point number by $\mathrm{fl}_{rnd} : \mathbb{R} \to \mathbb{F}$ for $rnd \in \{\square, \nabla, \Delta\}$, we denote by $rnd(\cdot)$ the result obtained when every operation in the expression within the parenthesis is executed in the specified rounding mode. If the order of execution is ambiguous such as in $C := \square(A \cdot B)$ for $A \in \mathbb{F}^{m\times k}$ and $B \in \mathbb{F}^{k\times n}$, then an assertion on C is true for any order of execution. The approximation $\square(A \cdot B)$ is to distinguish from the best approximation $\mathrm{fl}_\square(AB)$. Note that IEEE 754 defines for $a, b \in \mathbb{F}$, $\circ \in \{+, -, \cdot\}$ and $x := a \circ b \in \mathbb{R}$ scalar operations by $rnd(a \circ b) = \mathrm{fl}_{rnd}(x) \in \mathbb{F}$.

Thus $\nabla(A \cdot B)$ and $\triangle(A \cdot B)$ denote the result in rounding downwards and upwards, respectively, so that

$$\nabla(A \cdot B) \le AB \le \triangle(A \cdot B) \ . \tag{2.4}$$

[2]Sometimes the distance from 1.0 to the next *larger* floating-point number is used; for example, Matlab adopts this rule.

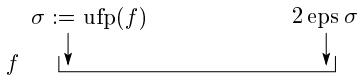

FIG. 2.1. *Normalized floating-point number: unit in the first place and unit in the last place*

Once the rounding is changed to a directed rounding, it need not to be changed again by using $\nabla(A \cdot B) = -\Delta((-A) \cdot B)$. This may be useful if changing the rounding mode is very time consuming.

To analyze conversion between infimum-supremum and midpoint-radius representation, we need the improved error estimates for floating-point operations I introduced in [41]. They are based on the "unit in the first place (ufp)" or leading binary bit of a real number, which is defined by

$$0 \neq r \in \mathbb{R} \quad \Rightarrow \quad \mathrm{ufp}(r) := 2^{\lfloor \log_2 |r| \rfloor} \ , \tag{2.5}$$

where $\mathrm{ufp}(0) := 0$. This concept is independent of a floating point format or underflow range, and it applies to real numbers. It gives a convenient way to characterize the bits of a normalized floating-point number f: they range between the leading bit $\mathrm{ufp}(f)$ and the unit in the last place $2\mathtt{eps}\,\mathrm{ufp}(f)$. In particular

$$f \in \mathbb{F} \quad \Rightarrow \quad f \in 2\mathtt{eps}\,\mathrm{ufp}(f)\,\mathbb{Z} \ , \tag{2.6}$$

which is also true in underflow, and

$$0 \neq x \in \mathbb{R} \ : \quad \mathrm{ufp}(x) \leq |x| < 2\mathrm{ufp}(x) \ . \tag{2.7}$$

The situation is illustrated in Figure 2.1, where the line depicts the bits of the floating-point representation of f. To the left is the leading bit $\mathrm{ufp}(f)$, which is the implicit 1 for normalized numbers. We often use

$$a \pm b \in 2\mathbb{U} \ \Rightarrow \ \mathrm{fl}(a \pm b) = a \pm b \qquad \text{and} \qquad x \notin 2\mathbb{U} \ \Rightarrow \ \mathrm{fl}_{rnd}(x)/2 \in \mathbb{F} \ \text{ for } rnd \in \{\square, \nabla, \Delta\} \ , \tag{2.8}$$

where $a, b \in \mathbb{F}$ and $x \in \mathbb{R}$. The first conclusion is known as Hauser's observation [10]. When rounding $a \circ b \in \mathbb{R}$ for $a, b \in \mathbb{F}$ and $\circ \in \{+, -, \cdot\}$ into $f := rnd(a \circ b) = \mathrm{fl}_{rnd}(a \circ b) \in \mathbb{F}$, the error is characterized [41] by

$$f = x + \delta + \eta \ \text{ with } \ |\delta| \leq \varphi\,\mathrm{ufp}(x) \leq \varphi\,\mathrm{ufp}(f) \leq \varphi|f|, \ |\eta| \leq \mathtt{eta}/2, \ \delta\eta = 0 \ , \tag{2.9}$$

where $\varphi = \mathtt{eps}$ for rounding to nearest and $\varphi = 2\mathtt{eps}$ for directed rounding. Using (2.8) we have for floating-point addition and subtraction

$$f = \mathrm{fl}_{\square}(a \pm b) \quad \Rightarrow \quad |f - (a \pm b)| \leq \mathtt{eps}\,\mathrm{ufp}(a \pm b) \leq \mathtt{eps}\,\mathrm{ufp}(f) \ , \tag{2.10}$$

$$f \in \{\mathrm{fl}_{\nabla}(a \pm b), \mathrm{fl}_{\Delta}(a \pm b)\} \quad \Rightarrow \quad |f - (a \pm b)| \leq 2\mathtt{eps}\,\mathrm{ufp}(a \pm b) \leq 2\mathtt{eps}\,\mathrm{ufp}(f) \ . \tag{2.11}$$

The second estimate (2.11) is particularly useful for Cell processors [15, 14].

Many interesting properties of $\mathrm{ufp}(\cdot)$ are given in [41] without which certain delicate estimates of errors in floating-point computations had hardly been possible. Another example is the following error estimate for dot products, which was proved in [40]. Note that the bound is computed in floating-point arithmetic.

THEOREM 2.1. *Let $A \in \mathbb{F}^{m \times k}$ and $B \in \mathbb{F}^{k \times n}$ with $2(k+2)\mathtt{eps} \leq 1$ be given, and let $C := \square(A \cdot B)$ and $\Gamma := \square(|A| \cdot |B|)$. Here C may be computed in any order, and we assume that Γ is computed in the same order. Then*

$$|\ \square(A \cdot B) - AB\ | \ \leq \ \square\big((k+2)\,\mathtt{eps}\,\mathrm{ufp}(\Gamma) + \mathtt{realmin}\big) \ . \tag{2.12}$$

The factor $k+2$ cannot be replaced by $k+1$.

REMARK 1. Usually the products C and Γ are computed by the same library routine, for example $xGEMM$ in BLAS3 [3, 8], so that it seems reasonable to assume that for matrices of the same size the operations are executed in the same order of execution.

REMARK 2. The result is true regardless of the order of execution of the two products in the right hand side of (2.12).

Fortunately there is a simple algorithm to compute the unit in the first place [41], which we repeat for convenience. It works correctly in the underflow range but causes overflow for input very close to $\pm$`realmax`. The latter case can be cured by scaling with a power of 2, however, at the cost of a branch.

ALGORITHM 2.2. *Unit in the first place of a floating-point number.*

$$\begin{array}{l} \text{function } S = \mathrm{ufp}(p) \\ \quad q = \mathrm{fl}(\varphi\, p) \qquad\qquad\qquad \text{for } \varphi := (2\mathtt{eps})^{-1} + 1 \\ \quad S = \mathrm{fl}(|q - (1 - \mathtt{eps})q|) \end{array}$$

Note that this algorithm can be applied to a matrix as well. For an $n \times n$-matrix, the unit in the first place is computed in $4n^2$ floating-point operations.

The classical bound [11, (3.5)], barring underflow, for the error of the floating-point matrix product is

$$|\Box(A \cdot B) - AB| \le \gamma_k |A||B| \quad \text{with} \quad \gamma_k := \frac{k\mathtt{eps}}{1 - k\mathtt{eps}}\ , \tag{2.13}$$

where here an in the following we implicitly assume $k\mathtt{eps} < 1$ when using γ_k. Estimate (2.12) is superior to the classical bound (2.13) because the ufp-concept and (2.7) imply that (2.12) is up to a factor 2 better than (2.13), whereas the latter is neither true when computed in rounding to nearest nor is it valid in the presence of underflow. On the other hand, (2.13) is independent of the sequence of operations because it depends on the real matrix product $|A||B|$. Both estimates do, in general, grossly overestimate the true error, for which the factor γ_k can usually be replaced by a small constant independent of k. Nevertheless, both estimates (2.12) and (2.13) are almost sharp in the worst case.

By (2.9), barring underflow, the classical bound (2.13) for rounding to nearest changes into

$$|\nabla(A \cdot B) - AB| \le \gamma_{2k} |A||B| \qquad \text{and} \qquad |\Delta(A \cdot B) - AB| \le \gamma_{2k} |A||B| \tag{2.14}$$

for directed rounding. For $\mathbf{A} = [A_1, A_2] \in \mathbb{R}^{m \times n}$ additional notations are

$$\mathrm{mid}([A_1, A_2]) := (A_1 + A_2)/2 \in \mathbb{R}^{m \times n} \qquad \text{and} \qquad \mathrm{rad}([A_1, A_2]) := (A_2 - A_1)/2 \in \mathbb{R}^{m \times n} \tag{2.15}$$

and

$$\mathbf{A} \in \mathbb{IR}^{m \times n}: \quad |\mathbf{A}| := \max\{|A| : A \in \mathbf{A}\} \tag{2.16}$$

where, as always, comparison between vectors and matrices is to be understood entrywise.

3. Conversion between infimum-supremum and midpoint-radius. Since we assume interval matrices to be given in infimum-supremum representation but, motivated by Table 1.1, want to use midpoint-radius representation, we need a conversion without sacrificing the inclusion principle. A given midpoint-radius representation $\langle mA, rA \rangle$ with $mA, rA \in \mathbb{F}^{n \times n}$ is easily converted into infimum-supremum representation by

$$A_1 := \nabla(mA - rA) \qquad \text{and} \qquad A_2 := \Delta(mA + rA)\ . \tag{3.1}$$

Obviously, $\langle mA, rA \rangle \subseteq [A_1, A_2]$. The optimality of operations with directed rounding (2.3) implies that the inclusion is best possible, but in general not with equality. The overestimation is estimated as follows.

THEOREM 3.1. *Let $mA, rA \in \mathbb{F}^{n\times n}$ be given and let A_1, A_2 be as in (3.1). Then*

$$rA \le \operatorname{rad}([A_1, A_2]) \le rA + 2\texttt{eps}\max\big(|mA|, |rA|\big) \ . \tag{3.2}$$

The factor 2 *is sharp up to* $\mathcal{O}(\texttt{eps})$.

PROOF. The left inequality is clear by $\langle mA, rA\rangle \subseteq [A_1, A_2]$. Since 2eps is the relative rounding error unit for directed rounding, (2.11) implies

$$\begin{aligned} |A_2 - A_1 - 2rA| &= |\Delta(mA + rA) - (mA + rA) - \nabla(mA - rA) + (mA - rA)| \\ &\le 2\texttt{eps}\big(|mA + rA| + |mA - rA|\big) \ , \end{aligned}$$

and the second inequality follows. The predecessor and successor of $1 + 2\texttt{eps}$ is 1 and $1 + 4\texttt{eps}$, respectively, so that $\langle 1 + 2\texttt{eps}, \texttt{eta}\rangle$ is transformed into $[1, 1 + 4\texttt{eps}]$ with radius 2eps. The assertion follows. □

If an entry of rA_{ij} is not too small relative to mA_{ij}, then the corresponding overestimation is small.

The midpoint-radius form allows, in contrast to the infimum-supremum form, to represent very narrow intervals. One might come to the conclusion that therefore the midpoint-radius form is preferable and should produce narrower inclusions in general. However, this is not true. In [38, Lemma 3.2] it is shown that under general assumptions the expected distance of $x \in \mathbb{R}$ to $\square(x)$ is of the order $\beta\texttt{eps}\,\operatorname{ufp}(x)$ with a factor β close to 1. This means that in general a relatively small radius disappears after the first interval operation, and there is no general benefit of the midpoint-radius representation in that respect.

In the conversion from infimum-supremum to midpoint-radius representation, the first problem is the choice of the midpoint. This is not as obvious as it may seem [9]. In particular, the midpoint should be inside the interval. For example, in a 2-digit decimal arithmetic, the obvious choice of the midpoint of $A := [0.62, 0.64]$ computes to $\Delta\big((0.62 + 0.64)/2\big) = 0.65$ and does not belong to A.

For a given infimum-supremum representation $[A_1, A_2]$ with $A_1, A_2 \in \mathbb{F}^{n\times n}$ we consider two ways of conversion into midpoint-radius representation, namely

$$mA' := \Delta(A_1 + (A_2 - A_1)/2) \qquad \text{and} \qquad rA' := \Delta(mA' - A_1) \tag{3.3}$$

and

$$mA := \Delta((A_1 + A_2)/2) \qquad \text{and} \qquad rA := \Delta(mA - A_1) \ . \tag{3.4}$$

From a performance point of view, the latter, which was proposed by Oishi [30], needs only $2n^2$ operations, whereas the computation of mA' requires $3n^2$ operations. But from a numerical point of view, the former is, in general, preferable to $(A_1 + A_2)/2$, cf. [4, 7].[3]

For the conversion from infimum-supremum to midpoint-radius representation in binary, however, the opposite is true. We will prove that (3.3) and (3.4) deliver in many cases the same midpoint and radius, but always $rA \le rA'$, and sometimes $rA < rA'$. For better readability we use a scalar interval $[a, b] \in \mathbb{IF}$ with $\mu := (a + b)/2$ and $\varrho := (b - a)/2$, $\mu, \varrho \in \mathbb{R}$, so that $[a, b] = \langle \mu, \varrho\rangle$. The following results hold for an interval matrix $[A_1, A_2]$ entrywise.

LEMMA 3.2. *Let $a, b \in \mathbb{F}$, $a \le b$ be given and define $m', r', m, r \in \mathbb{F}$ by*

$$m' := \Delta(a + (b - a)/2) \qquad \textit{and} \qquad r' := \Delta(m' - a) \tag{3.5}$$

and

$$m := \Delta((a + b)/2) \qquad \textit{and} \qquad r := \Delta(m - a) \ . \tag{3.6}$$

[3] In decimal arithmetic (3.4) is not even acceptable because the computed midpoint mA may not be in $[A_1, A_2]$.

Then

$$[a,b] \subseteq \langle m',r' \rangle \qquad \textit{and} \qquad [a,b] \subseteq \langle m,r \rangle \ . \tag{3.7}$$

Furthermore,

$$m = \mathrm{fl}_\Delta\big(\frac{a+b}{2}\big) \ . \tag{3.8}$$

Remark. Note that (3.8) is not true for m'. In fact, m' may differ significantly from m:

$$[-1+\mathtt{eps},1] \ \text{ is transformed into } \ \langle m,r \rangle = \langle \mathtt{eps}/2,1 \rangle \ , \ \text{but } \langle m',r' \rangle = \langle \mathtt{eps},1 \rangle \ , \tag{3.9}$$

so that $m = (a+b)/2$ but $m' = 2m$.

Proof. The rounding mode implies $m \geq \mu$ and $m' \geq \mu$. For any $\mu \leq \hat{m} \in \mathbb{F}$ and $\hat{r} := \Delta(\hat{m} - a)$ it follows $\hat{r} \geq \hat{m} - a \geq \mu - a = b - \mu$ and

$$\hat{m} - \hat{r} \leq a \leq b = \mu + b - \mu \leq \hat{m} + \hat{r} \ ,$$

which means $[a,b] \subseteq \langle \hat{m},\hat{r} \rangle$ and therefore (3.7). If $a+b \in 2\mathbb{U}$, then (2.8) implies $a+b \in \mathbb{F}$ and (3.8) follows. If $a+b \notin 2\mathbb{U}$, then (2.8) implies $m = \Delta(s/2) = s/2$ for $s := \Delta(a+b)$, so that (3.8) follows again. □

The rounding upwards in (3.5) and (3.6) cause a certain unsymmetry; therefore we need Sterbenz' lemma for $a,b \geq 0$ and for $a,b \leq 0$.

Lemma 3.3. *If*

$$0 \leq \frac{a}{2} \leq b \leq 2a \qquad \textit{or} \qquad 2b \leq a \leq \frac{b}{2} \leq 0 \ , \tag{3.10}$$

then $b - a \in \mathbb{F}$.

Proof. If $b - a \in \mathbb{U}$, then (2.8) implies $b - a \in \mathbb{F}$. Otherwise, this is Sterbenz' lemma [42] for the first assumption in (3.10), and again Sterbenz' lemma implies $|a| - |b| = b - a \in \mathbb{F}$ for the second assumption. □

Lemma 3.4. *Let* $a,b \in \mathbb{F}$, $a \leq b$ *be given and define* $m',r',m,r \in \mathbb{F}$ *by (3.5) and (3.6). Then (3.10) implies* $m' = m$ *and* $r' = r$.

Proof. Assume $b - a \in 2\mathbb{U}$. Then $b - a \in \mathbb{F}$ by (2.8) and $m = \Delta(\mu)$ by (3.8). By $\mathbb{F} \subseteq \mathtt{eta}\mathbb{Z}$, $a+b$ and $a-b$ are either both even or both odd multiples of $\mathtt{eta}$. In the first case $(b-a)/2 \in \mathbb{F}$ and $m' = \Delta(\mu) = m$. Otherwise, $\Delta\big((b-a)/2\big) = (b-a)/2 + \mathtt{eta}/2$, and

$$m' = \Delta\big(a + (b-a)/2\big) = \mathrm{fl}_\Delta(\mu + \mathtt{eta}/2) = \mathrm{fl}_\Delta(\mu) = m \ .$$

If $a+b \in 2\mathbb{U}$, then the definition (2.1) of $\mathbb{U}$ and $ab \geq 0$ imply $a, b, b-a \in 2\mathbb{U}$, and we may proceed as before. Now suppose $a+b, b-a \notin 2\mathbb{U}$. Hence Lemma 3.3 implies $b-a \in \mathbb{F}$, and $b-a \notin 2\mathbb{U}$ together with (2.8) yields $\Delta\big((b-a)/2\big) = (b-a)/2$. It follows $m' = \Delta(a + (b-a)/2) = \mathrm{fl}_\Delta((a+b)/2) = m$, and therefore $r' = r$. □

As has been mentioned, $m := \Delta\big((a+b)/2\big)$ may be outside the interval $[a,b]$ in decimal arithmetic.[4] This cannot happen in binary arithmetic for both m and m' computed by (3.5) and (3.6), respectively. Moreover, the radius r as computed by (3.6) is never larger than r' in (3.5).

Theorem 3.5. *Let* $a,b \in \mathbb{F}$, $a \leq b$ *be given and define* $m',r',m,r \in \mathbb{F}$ *by (3.5) and (3.6). Then*

$$m', m \in [a,b] \qquad \textit{and} \qquad r' \geq r \ , \tag{3.11}$$

but not necessarily $\langle m,r \rangle \subseteq \langle m',r' \rangle$.

[4] This is also true for $\square((a+b)/2)$ as for the same example $[0.62, 0.64]$ in 2-digit decimal arithmetic.

PROOF. By (3.8), $m = \mathrm{fl}_\Delta(\frac{a+b}{2}) \in [a,b]$. By Lemma 3.4, $m' \neq m$ can happen only for wide intervals, and a case distinction proves $m' \in [a,b]$. Since $m = \mathrm{fl}_\Delta\big((a+b)/2\big) = \min\{f \in \mathbb{F} : (a+b)/2 \leq f\}$ and $\mu \leq m'$, it follows $m \leq m'$ and therefore $r' \geq r$. The last assertion follows by the example in (3.9). □

The question arises whether there is an optimal conversion into midpoint-radius representation, and whether it is computable in floating-point. Both is true, shown by the following lemma.

THEOREM 3.6. *Let $a, b \in \mathbb{F}$, $a \leq b$ be given and define*

$$M := \square((a+b)/2) \qquad \text{and} \qquad R := \max\big(\Delta(M-a), \Delta(b-M)\big)\ . \tag{3.12}$$

Then $[a,b] \subseteq \langle M, R\rangle$. Abbreviate $\mu := \frac{a+b}{2}$ and $\varrho := \frac{b-a}{2}$. The midpoint M and radius R are optimal as by

$$|M-\mu| \leq |m-\mu| \qquad \text{and} \qquad R \leq r \tag{3.13}$$

for any $m, r \in \mathbb{F}$ with $[a,b] \subseteq \langle m, r\rangle$. Nevertheless, $\langle M, R\rangle \subseteq \langle m, r\rangle$ is not necessarily true.

REMARK. One might use $R' := \Delta(0.5 \cdot (b-a))$ to minimize the radius and $M' := \Delta(a + R')$ as proposed in [27]. Indeed one can show $R' = \mathrm{fl}_\Delta(\varrho)$, however, not necessarily $[a,b] \subseteq \langle M', R'\rangle$ as by $[a,b] := [1, 1 + 2\mathtt{eps}]$. In that example $R' = \mathtt{eps} = \varrho$ is the true radius, however, there is no floating-point midpoint m with $[a,b] \subseteq \langle m, R'\rangle$.

PROOF. The rounding upwards in the computation of R implies $R \geq \max(M-a, b-M)$, so that

$$M - R \leq M - (M-a) = a \leq b = M + (b-M) \leq M + R\ .$$

If $a+b \in 2\mathbb{U}$, then (2.8) implies $a+b \in \mathbb{F}$ and therefore $M = \square(\mu)$, and if $a+b \notin 2\mathbb{U}$, then $M = \square(s/2) = s/2$ for $s := \square(a+b)$ again by (2.8). This proves $M = \square(\mu)$, and therefore $|M-\mu| \leq |m-\mu|$ for any $m \in \mathbb{F}$.

Suppose $[a,b] \subseteq \langle m, r\rangle$, then $m - r \leq a \leq b \leq m + r$ and (2.2) imply

$$r \geq \max(m-a, b-m) = \max(m-\mu, \mu-m) + \varrho = |\mu - m| + \varrho \geq |\mu - M| + \varrho = \max(M-a, b-M)\ .$$

The definition of R, the optimality (2.2) of the upward rounding and $r \in \mathbb{F}$ prove $r \geq R$. Finally, the interval $[1 - \mathtt{eps}, 1 + 2\mathtt{eps}]$ is converted into $\langle M, R\rangle = \langle 1, 2\mathtt{eps}\rangle$ and $\langle m, r\rangle = \langle 1 + 2\mathtt{eps}, 3\mathtt{eps}\rangle$. □

Although $\langle M, R\rangle$ is never inferior to $\langle m, r\rangle$ defined by (3.6), the practical difference is marginal. In 10^9 random test cases $[a,b]$ with radii ranging from very narrow to very wide, the median of the relative errors $(r-R)/R$ between r and R was zero, and the mean below `eps`. Hence one may save the additional computational effort in (3.12).

All possibilities (3.5), (3.6) and (3.12) have their drawbacks with unexpected overflow. The interval

$$[-\mathtt{realmax}, \mathtt{realmax}] \qquad \text{is converted into} \qquad \langle m', r'\rangle = \langle \infty, \infty\rangle \quad \text{and} \quad \langle m, r\rangle = \langle 0, \mathtt{realmax}\rangle\ ,$$

whereas

$$[\mathtt{realmax}, \mathtt{realmax}] \qquad \text{is converted into} \qquad \langle m', r'\rangle = \langle \mathtt{realmax}, 0\rangle \quad \text{and} \quad \langle m, r\rangle = \langle \infty, \infty\rangle\ ,$$

and $\langle M, R\rangle = \langle m, r\rangle$ in both cases. One might want to cure the second problem by defining $\hat{m} := \Delta\big(a/2 + b/2\big)$. Then, however, $\hat{m} = 2\mathtt{eta} \notin [a,b] := [\mathtt{eta}, \mathtt{eta}]$, and also $M := \square\big(a/2 + b/2\big) = 0 \notin [a,b]$.

We have already seen in (3.9) that the midpoints m' and m may differ significantly, while $m = \Delta(\mu)$ and $M = \square(\mu)$ cannot. The radius r' may at least be larger by two units in the last place than r as for $[a,b] := [1 + 6\mathtt{eps}, 7 + 40\mathtt{eps}]$ with

$$\langle m', r'\rangle = \langle 4 + 32\mathtt{eps}, 3 + 28\mathtt{eps}\rangle \quad \text{but} \quad \langle m, r\rangle = \langle 4 + 24\mathtt{eps}, 3 + 20\mathtt{eps}\rangle\ .$$

By Lemma 3.4, however, $r' > r$ can only happen for intervals with large diameter. Summarizing, possibilities (3.5), (3.6) and (3.12) have their advantages and disadvantages. Henceforth we use (3.6) (and (3.4) for matrices) because it requires less operations and always $r \le r'$. Another reason are the nice, forthcoming estimates (3.16) in Theorem 3.8, which are weaker for $\langle m', r' \rangle$.

The conversion from infimum-supremum to midpoint-radius representation causes an inevitable overestimation if the midpoint $\mu = (a+b)/2$ is not a floating-point number (e.g. if a and b differ only in the last bit). The amount of overestimation for not too wide intervals is estimated as follows.

LEMMA 3.7. *Let $a, b \in \mathbb{F}$, $a \le b$ be given and define $m, r \in \mathbb{F}$ by (3.6). If*

$$0 \le a \le b \le 3a \qquad \text{or} \qquad 3b \le a \le b \le 0 \ , \tag{3.14}$$

then $m - a \in \mathbb{F}$, i.e. the radius r is computed without rounding error. If in addition $a + b \notin 2\mathbb{U}$, then

$$0 \le m - \mu = r - \varrho \le 2\mathtt{eps}\,\mathrm{ufp}(\mu) \tag{3.15}$$

for $\mu := \frac{a+b}{2}$ and $\varrho := \frac{b-a}{2}$.

REMARK. The assumption $a + b \notin 2\mathbb{U}$ for (3.15) is necessary as for $[a, b] := [0, \mathtt{eta}]$ with $m = r = \mathtt{eta}$ but $\mu = \varrho = \mathtt{eta}/2$, so that $m - \mu = r - \varrho = 1 \cdot \mathrm{ufp}(\mu)$.

PROOF. If the first assumption in (3.14) is true, then $a \le \mu = \frac{a+b}{2} \le 2a \in \mathbb{F}$, and by (3.8) and by the directed rounding in (3.6) it follows

$$\frac{1}{2}a \le a \le \mu \le \Delta(\mu) = m \le 2a \ ,$$

where the last inequality uses that $\alpha \in \mathbb{R}$ and $\alpha \le f \in \mathbb{F}$ implies $\mathrm{fl}_\Delta(\alpha) \le f$. Assume the second assumption in (3.14) is true, so that $m \le 0$. If $a + b \in 2\mathbb{U}$, then also $0 \ge a, b, m, m - a \in 2\mathbb{U}$ and therefore $m - a \in \mathbb{F}$. If $a + b \notin 2\mathbb{U}$, then

$$2m = 2\Delta\big(\frac{a+b}{2}\big) = \Delta(a+b) \le \mathrm{fl}_\Delta(a) = a \le \frac{a+b}{2} = \mu \le m \le \frac{m}{2} \ .$$

Thus by Lemma 3.3, any of the two assumptions in (3.14) implies $m - a \in \mathbb{F}$ and therefore $r = m - a$, so that $r - \varrho = m - a - \varrho = m - \mu$. Finally (2.11) implies $\Delta(a+b) \le a + b + 2\mathtt{eps}\,\mathrm{ufp}(a+b)$ and $m = \Delta\big((a+b)/2\big) = \Delta(a+b)/2$ since $a + b \notin 2\mathbb{U}$, and the assertion follows by $\mathrm{ufp}(a+b) = 2\mathrm{ufp}\big((a+b)/2\big) = 2\mathrm{ufp}(\mu)$. □

The amount of overestimation for general intervals is estimated as follows. The proof is lengthy and deferred to the appendix.

THEOREM 3.8. *Let $a, b \in \mathbb{F}$, $a \le b$ be given and define $m, r \in \mathbb{F}$ by (3.6), $\mu := (a+b)/2$ and $\varrho := (b-a)/2$. If $\mathtt{eps} \le \frac{1}{4}$ and no underflow occurs in the computation of m and r, then*

$$0 \le m - \mu \le 2\mathtt{eps}\,\mathrm{ufp}(\mu) \qquad \text{and} \qquad 0 \le r - \varrho \le 2\mathtt{eps}\big(|\mu| + \varrho\big) \ . \tag{3.16}$$

REMARK. The estimates are not true for m', r' as defined in (3.5). Consider $[a, b] := [-2 + 2\mathtt{eps}, 2 + 4\mathtt{eps}]$, then $\mu = 3\mathtt{eps}$ and $m' = 6\mathtt{eps}$, so that

$$m' - \mu = 3\mathtt{eps} = 1.5\mathtt{eps}^{-1} \cdot \mathtt{eps}\,\mathrm{ufp}(\mu) \ ,$$

and for $[a, b] := [-1 + 3\mathtt{eps}, 3 + 4\mathtt{eps}]$ we have $\mu = 1 + 3.5\mathtt{eps}$, $\varrho = 2 + 0.5\mathtt{eps}$, $m' = 1 + 8eps$, $r' = 2 + 8\mathtt{eps}$, and therefore

$$r' - \varrho = 7.5\mathtt{eps} \approx 2.5 \cdot \mathtt{eps}\big(|\mu| + \varrho\big) \ .$$

This means that the first factor 2 in (3.16) must be replaced by a much larger number to bound $m' - \mu$, and the second factor 2 for $r' - \varrho$ at least by a number close to 2.5.

COROLLARY 3.9. *Let* $\boldsymbol{A} = [A_1, A_2] \in \mathbb{F}^{n\times n}$ *be given and define* $M, R \in \mathbb{F}^{n\times n}$ *by*

$$M := \Delta((A_1 + A_2)/2) \qquad \text{and} \qquad R := \Delta(M - A_1)\ .$$

If no underflow occurs in the computation of M *and* R*, then*

$$0 \le M - \mathrm{mid}(\boldsymbol{A}) \le 2\mathtt{eps}\,\mathrm{ufp}(\mathrm{mid}(\boldsymbol{A})) \qquad \text{and} \qquad 0 \le R - \mathrm{rad}(\boldsymbol{A}) \le 2\mathtt{eps}\,|\boldsymbol{A}| \tag{3.17}$$

for $\mathrm{mid}(\boldsymbol{A}), \mathrm{rad}(\boldsymbol{A}), |\boldsymbol{A}| \in \mathbb{R}^{n\times n}$ *as defined in (2.15) and (2.16).*

4. Matrix multiplication. We are aiming on fast implementations of interval matrix multiplication. To simplify the exposition, we explain the methods for square matrices; the generalization to rectangular matrices is straightforward. Let $A, B \in \mathbb{F}^{n\times n}$ and $\mathbf{A}, \mathbf{B} \in \mathbb{IF}^{n\times n}$ be given, then three cases are distinguished:

$$\begin{array}{lll} 1) & A \cdot B & \text{two point matrices to interval result} \\ 2) & A \cdot \mathbf{B} & \text{point matrix times interval matrix} \\ 3) & \mathbf{A} \cdot \mathbf{B} & \text{two interval matrices} \end{array} \tag{4.1}$$

4.1. Two point matrices to interval result. The first case is directly solved by (2.4). An implementation in executable INTLAB code is as follows.

ALGORITHM 4.1. *Product of two floating-point matrices to interval result.*

```
function C = FFmul(A,B)
  setround(-1)              % switch rounding to downwards
  Cinf = A*B;               % floating-point product rounded downwards
  setround(1)               % switch rounding to upwards
  Csup = A*B;               % floating-point product rounded upwards
  C = infsup(Cinf,Csup);    % compose result of infimum and supremum
```

The algorithm is correct, also in the presence of underflow, and there is no restriction on the dimension n. Algorithm 4.1 requires 2 matrix multiplications. There is no performance improvement using the a priori estimate (2.12) because it requires the matrix product $|A||B|$. It can be used through estimates of the maximum absolute value of elements in the rows of A and columns of B, cf. [31]. However, this introduces an additional dependency on the size of the entries of A and B.

Algorithm 4.1 produces the same result as the classical approach (1.5), but there is an overestimation compared to the best possible inclusion. The overestimation is proportional to the condition number of the individual dot products: Let $x, y \in \mathbb{F}^n$ be given with $c := x^T y \neq 0$, and assume that no underflow occurs. Then by (2.9) the radius ϱ of the narrowest inclusion $\mathbf{c} := [\mathrm{fl}_\nabla(c), \mathrm{fl}_\Delta(c)] \in \mathbb{IF}$ satisfies $\varrho \le \mathtt{eps}|x^T y|$, and by (2.13) the radius r of the inclusion `C` computed by Algorithm 4.1 (and of the classical approach (1.5)) satisfies $r \le \gamma_n |x^T||y|$. Hence the overestimation is bounded by

$$r \le \frac{n}{1 - n\mathtt{eps}}\ \mathrm{cond}(x^T y)\,\varrho\ , \tag{4.2}$$

using the condition number

$$\mathrm{cond}(x^T y) = \frac{|x^T||y|}{|x^T y|} = \lim_{\varepsilon\to 0} \sup\big\{ \big| \frac{(x + \Delta x)^T (y + \Delta y) - x^T y}{2\varepsilon x^T y} \big| : |\Delta x| \le \varepsilon|x|, |\Delta y| \le \varepsilon|y| \big\}\ .$$

A more accurate inclusion of the precise result $AB \in \mathbb{R}^{n\times n}$ can be computed using accurate dot product techniques as described in [20, 41].

4.2. Point matrix times interval matrix. It is known [28] that for matrices $A \in \mathbb{R}^{n\times n}, \mathbf{B} = [B_1, B_2] = \langle mB, rB\rangle \in \mathbb{IR}^{n\times n}$ with *real* endpoints

$$\mathbb{PR}(A\mathbf{B}) = \{AB : B \in \mathbf{B}\} = \langle\ AmB\ ,\ |A|rB\ \rangle\ . \tag{4.3}$$

The conversion of $[B_1, B_2]$ into midpoint-radius representation is performed by (3.6), so that an inclusion $[C_1, C_2]$ of $\mathbb{PR}(A\mathbf{B})$ can be computed as follows.

Algorithm 4.2. *Point matrix times interval matrix, requires* 3 *point matrix multiplications.*

function $[\mathrm{C}_1, \mathrm{C}_2] = \mathrm{FImul}_3(\mathrm{A}, \boldsymbol{B})$
$mB := \triangle\big((B_1 + B_2)/2\big)$
$rB := \triangle\big(mB - B_1\big)$ % $[B_1, B_2] \subseteq \langle mB, rB\rangle$
$rC := \triangle\big(|A| \cdot rB\big)$ % $|A| \cdot rB \le rC$
$C_2 := \triangle\big(A \cdot mB + rC\big)$ % $A \cdot mB + |A| \cdot rB \le C_2$
$C_1 := \nabla\big(A \cdot mB - rC\big)$ % $C_1 \le A \cdot mB - |A| \cdot rB$

As by Lemma 3.2, the algorithm is correct, also in the presence of underflow, there is no restriction on the dimension n, and it requires 3 matrix multiplications.

For the analysis of the classical approach (1.5) and Algorithm 4.2, let $x \in \mathbb{F}^n$ and $\mathbf{y} = [y_1, y_2] = \langle \mu, \varrho\rangle \in \mathbb{IF}^n$ with $y_1, y_2 \in \mathbb{F}^n$, $\mu, \varrho \in \mathbb{R}^n$ be given. Note that entries of x and of the infimum and supremum of $\mathbf{y}$ are floating-point numbers, whereas μ and ϱ are real quantities and will be used only for the analysis. As in (4.3), the power set dot product (1.3) is

$$\mathbb{PR}(x^T\mathbf{y}) = \{x^T y : y \in \mathbf{y}\} = \langle x^T\mu, |x^T|\varrho\rangle = [x^T(\mu - S\varrho), x^T(\mu + S\varrho)]\ , \tag{4.4}$$

where $S \in \mathbb{R}^{n\times n}$ is the diagonal matrix with $S_{ii} = \mathrm{sign}(x_i)$. Note that the entries of $r_1 := \mu - S\varrho$ and $r_2 := \mu + S\varrho$ are entries of y_1 or y_2, so that $r_1, r_2 \in \mathbb{F}^n$. Those entries are selected by the classical approach (1.5) via a case distinction. This means $\mathrm{rad}(\mathbb{PR}(x^T\mathbf{y})) = |x^T|\mathrm{rad}(\mathbf{y}) = |x^T|\varrho = x^T r_2 - x^T r_1$ and

$$\mathit{classical}(x^T \cdot \mathbf{y}) = [\nabla(x^T r_1), \Delta(x^T r_2)]\ . \tag{4.5}$$

Barring underflow, (2.14) gives

$$|\nabla(x^T r_1) - x^T r_1| \le \gamma_{2n}|x^T||r_1| \qquad \text{and} \qquad |\Delta(x^T r_2) - x^T r_2| \le \gamma_{2n}|x^T||r_2|\ .$$

The i-th entry of $|r_1| + |r_2|$ is $|\mu_i - \varrho_i| + |\mu_i + \varrho_i| = 2\max(|\mu_i|, |\varrho_i|)$, so that

$$\mathrm{rad}\big(\mathit{classical}(x^T \cdot \mathbf{y})\big) \le |x^T|\mathrm{rad}(\mathbf{y}) + \gamma_{2n}|x^T|\max(|\mu|, \varrho)\ . \tag{4.6}$$

Applied to the interval matrix multiplication this means

$$\mathrm{rad}\big(\mathit{classical}(A \cdot \mathbf{B})\big) \le |A|\mathrm{rad}(\mathbf{B}) + \gamma_{2n}|A|\max(|\mathrm{mid}(\mathbf{B})|, \mathrm{rad}(\mathbf{B}))\ . \tag{4.7}$$

When using Algorithm 4.2, an additional overestimation occurs by the conversion of $[B_1, B_2]$ into $\langle mB, rB\rangle$. For the analysis barring underflow use Corollary 3.9 to see

$$|mB| \le (1 + 2\mathtt{eps})\mathrm{mid}(\mathbf{B}) \qquad \text{and} \qquad |rB| \le \mathrm{rad}(\mathbf{B}) + 2\mathtt{eps}|\mathbf{B}|\ . \tag{4.8}$$

Abbreviating $P := \Delta(A \cdot mB)$, (2.14) yields $|P - A \cdot mB| \le \gamma_{2n}|A||mB|$, so that

$$\begin{aligned} |C_2 - (A \cdot mB + rC)| &= |C_2 - (P + rC) + (P - A \cdot mB)| \\ &\le 2\mathtt{eps}|P + rC| + \gamma_{2n}|A||mB| \\ &\le 2\mathtt{eps}(1 + \gamma_{2n})|A||mB| + 2\mathtt{eps}\,rC + \gamma_{2n}|A||mB| \\ &\le \gamma_{2n+2}|A|\mathrm{mid}(\mathbf{B}) + 2\mathtt{eps}\,rC\ , \end{aligned} \tag{4.9}$$

and similarly

$$|C_1 - (A \cdot mB - rC)| \le \gamma_{2n+2}|A|\mathrm{mid}(\mathbf{B}) + 2\mathtt{eps}\, rC \ . \tag{4.10}$$

Furthermore, (2.14) and (4.8) yield

$$rC \le (1+\gamma_{2n})|A|rB \le (1+\gamma_{2n})|A|\big(\mathrm{rad}(\mathbf{B}) + 2\mathtt{eps}|\mathbf{B}|\big) \ ,$$

so that with a little computation

$$\begin{array}{rcl} \mathrm{rad}([C_1,C_2]) & \le & rC + \gamma_{2n+2}|A|\mathrm{mid}(\mathbf{B}) + 2\mathtt{eps}\, rC \\ & \le & (1+2\mathtt{eps})(1+\gamma_{2n})|A|\big(\mathrm{rad}(\mathbf{B}) + 2\mathtt{eps}|\mathbf{B}|\big) + \gamma_{2n+2}|A|\mathrm{mid}(\mathbf{B}) \\ & \le & |A|\mathrm{rad}(\mathbf{B}) + 2\mathtt{eps}(1+2\mathtt{eps})(1+\gamma_{2n})|A||\mathbf{B}| + \gamma_{2n+2}|A||\mathbf{B}| \\ & \le & |A|\mathrm{rad}(\mathbf{B}) + \gamma_{2n+4}|A||\mathbf{B}| \ . \end{array} \tag{4.11}$$

In theory, (4.3) implies that without rounding errors the classical product based on (1.5) and the result of Algorithm 4.2 is identical with the power set product $\mathbb{PR}(\mathbf{AB})$. In practice, Algorithm 4.2 uses the computed $\langle B_1, B_2\rangle$ rather than $[B_1, B_2]$, and due to this additional overestimation one might conclude that Algorithm 4.2 cannot be superior to the classical interval matrix multiplication, i.e. $classical(A \cdot \mathbf{B}) \subseteq [C_1, C_2]$.

As already pointed out in [36], this is not true. First, we generate random matrices $A, B \in \mathbb{F}^{n\times n}$ with pseudo-random entries drawn from a normal distribution with mean zero and standard deviation one, and define $\mathbf{B} := B \cdot [1-e, 1+e]$. Second, we use the Matlab function `randsvd` to generate a random matrix B with $\mathrm{cond}(B) = 10^{10}$, set A to be the Matlab-approximation `inv(B)` of B^{-1} and $\mathbf{B} := B \cdot [1-e, 1+e]$. Results are displayed in Table 4.1, where a ratio less than 1 means that the inclusion computed by $FImul_3$ is narrower by that factor than the classical bound. In both test sets there is practically no difference between

TABLE 4.1
Minimum, mean, median and maximum ratio of the radii of the results of $FImul_3(A, \boldsymbol{B})$ and $classical(A \cdot \boldsymbol{B})$, $n = 100$.

	well-conditioned				ill-conditioned			
e	minimum	mean	median	maximum	minimum	mean	median	maximum
10^{-11}	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
10^{-12}	0.9998	1.0000	1.0000	1.0003	0.9998	1.0000	1.0000	1.0001
10^{-13}	0.9977	1.0007	1.0006	1.0032	0.9994	1.0006	1.0006	1.0020
10^{-14}	0.9846	1.0061	1.0058	1.0365	0.9953	1.0052	1.0053	1.0163
10^{-15}	0.9298	1.0054	1.0022	1.1000	0.9428	1.0001	1.0000	1.0589
10^{-16}	0.9375	1.1096	1.1087	1.2750	0.9763	1.1339	1.1353	1.2753

$classical(\mathbf{A} \cdot \mathbf{B})$ and $FImul_3(A, \mathbf{B})$ for larger values of e, but the situation changes for small values of e, i.e. narrow intervals. The classical interval matrix multiplication is, in general, superior to Algorithm 4.2, but sometimes the inevitable presence of rounding errors helps Algorithm 4.2 to produce the narrower inclusion. The tests are performed with $n = 100$, for other dimensions the situation is similar. An explicit example for that phenomenon is given in [36, (16)].

Another test generates badly scaled A and $\mathbf{B}$ using (in INTLAB notation)

$$\begin{array}{l} \mathtt{A} = (\mathtt{sign}(\mathtt{randn}(\mathtt{n})) .* \mathtt{exp}(\mathtt{f} * \mathtt{randn}(\mathtt{n}))); \\ \mathtt{B} = (\mathtt{sign}(\mathtt{randn}(\mathtt{n})) .* \mathtt{exp}(\mathtt{f} * \mathtt{randn}(\mathtt{n}))) .* \mathtt{midrad}(1, \mathtt{e}); \end{array} \tag{4.12}$$

for values $f = 5$ and $f = 30$. As can be seen in Table 4.2, there is not too much difference to the first test set. The smallest value $e = 10^{-16}$ produces narrow entries $\mathbf{B}_{ij}$ with bounds differing in 2 bits or so. Then the maximum ratio 1.6667 occurred when the width of the result of $FImul_3$ was 5 bits, whereas that of the classical method was 3 bits.

TABLE 4.2
Minimum, mean, median and maximum ratio of the radii of the results of $FImul_3(A, \boldsymbol{B})$ and $classical(A \cdot \boldsymbol{B})$, $n = 100$.

	wide range $f = 5$				very wide range $f = 30$			
e	minimum	mean	median	maximum	minimum	mean	median	maximum
10^{-11}	0.9999	1.0000	1.0000	1.0001	1.0000	1.0000	1.0000	1.0000
10^{-12}	0.9996	1.0001	1.0000	1.0007	0.9998	1.0001	1.0000	1.0004
10^{-13}	0.9964	1.0011	1.0012	1.0060	0.9980	1.0012	1.0013	1.0063
10^{-14}	0.9806	1.0086	1.0089	1.0400	0.9828	1.0096	1.0100	1.0417
10^{-15}	0.9091	1.0141	1.0042	1.2222	0.8750	1.0188	1.0030	1.1429
10^{-16}	0.9545	1.0549	1.0417	1.6667	0.9444	1.0547	1.0345	1.6667

One matrix multiplication may be saved by using Theorem 2.1 to estimate the error in the floating-point computation of $A \cdot mB$. The algorithm is as follows.

ALGORITHM 4.3. *Point matrix times interval matrix, requires* 2 *point matrix multiplications.*

function $[\mathrm{C}_1, \mathrm{C}_2] = \mathrm{FImul}_2(\mathrm{A}, \boldsymbol{B})$

$mB := \triangle\big((B_1 + B_2)/2\big)$

$rB := \triangle\big(mB - B_1\big)$ % $[B_1, B_2] \subseteq \langle mB, rB\rangle$

$mC := \square\big(A \cdot mB\big)$ % *floating-point approximation*

$rC := \triangle\big(|A| \cdot \big[(n+2)\mathtt{eps}|mB| + rB\big] + \mathtt{realmin}\big)$ % $|\square(mC - A \cdot B)| + |A| \cdot rB \le rC$

$C_2 := \triangle\big(mC + rC\big)$ % $A \cdot mB + |A| \cdot rB \le C_2$

$C_1 := \nabla\big(mC - rC\big)$ % $C_1 \le A \cdot mB - |A| \cdot rB$

The algorithm is correct, also in the presence of underflow, provided $2(n+2)\mathtt{eps} \le 1$. Algorithm 4.3 requires 2 matrix multiplications. The proof of validity follows by Theorem 2.1, by

$$\mathrm{ufp}\big(\square(|A| \cdot |mB|)\big) \le \square(|A| \cdot |mB|) \le \triangle(|A| \cdot |mB|) \tag{4.13}$$

and the proper choice of rounding modes. If no underflow occurs and ignoring `realmin`, the radius rC computed by Algorithm 4.3 satisfies

$$rC \le (1 + \gamma_{2n})|A|(1 + \mathtt{eps})\big[(1 + \mathtt{eps})(n+2)\mathtt{eps}|mB| + rB\big] \ ,$$

where the first factor $1 + \mathtt{eps}$ stems from the addition and the second from the multiplication by $(n+2)$ in $\big[(n+2)\mathtt{eps}|mB| + rB\big]$ (multiplication by `eps` causes no rounding error). The estimates (4.9) and (4.10) remain valid, so that similar to (4.11) we obtain

$$\begin{aligned} \mathrm{rad}([C_1, C_2]) &\le (1 + 2\mathtt{eps})rC + \gamma_{2n+2}|A|\mathrm{mid}(\mathbf{B}) \\ &\le |A|\mathrm{rad}(\mathbf{B}) + \gamma_{2n+3}|A|\mathrm{rad}(\mathbf{B}) + \gamma_3|A||\mathbf{B}| + \gamma_{3n+6}|A|\mathrm{mid}(\mathbf{B}) \\ &\le |A|\mathrm{rad}(\mathbf{B}) + \gamma_{3n+9}|A||\mathbf{B}| \end{aligned} \tag{4.14}$$

for $[C_1, C_2]$ computed by Algorithm 4.3. By (4.3), the radius of the power set product $\mathbb{PR}(A\mathbf{B})$ is $|A|\mathrm{rad}(\mathbf{B})$, so that using (4.7), (4.11) and (4.14) we can summarize the results as follows.

THEOREM 4.4. *Let $A \in \mathbb{F}^{n\times n}$ and $\boldsymbol{B} \in \mathbb{IF}^{n\times n}$ be given. If no underflow occurs and ignoring* `realmin` *in Algorithm 4.3, then*

$$\begin{aligned} \mathrm{rad}\big(\mathbb{PR}(A\boldsymbol{B})\big) &= |A|\mathrm{rad}(\boldsymbol{B}) \ , \\ \mathrm{rad}\big(classical(A \cdot \boldsymbol{B})\big) &\le |A|\mathrm{rad}(\boldsymbol{B}) + \gamma_{2n}|A|\max(|\mathrm{mid}(\boldsymbol{B})|, \mathrm{rad}(\boldsymbol{B})) \ , \\ \mathrm{rad}\big(Algo\ 4.2\big) &\le |A|\mathrm{rad}(\boldsymbol{B}) + \gamma_{2n+4}|A||\boldsymbol{B}| \\ \mathrm{rad}\big(Algo\ 4.3\big) &\le |A|\mathrm{rad}(\boldsymbol{B}) + \gamma_{3n+9}|A||\boldsymbol{B}| \end{aligned}$$

is true, where $\mathrm{rad}\big(Algo4.x\big)$ *denotes the radius of the result* $[C_1, C_2] \in \mathbb{IF}^{n\times n}$ *computed by Algorithm* $\mathtt{FImul}_x$.

Remark. Note that Theorem 4.4 states *estimates* of the radii, based on the standard estimates (2.14) of the error of a matrix product. The latter are essentially sharp, but do in general grossly overestimate the true error. So the estimates should, if ever, be compared with care. Having said this, we see that the estimates are of a similar quality provided the interval entries are not too wide; the additional conversion of $[B_1, B_2]$ into midpoint-radius representation is hardly visible.

4.3. Two interval matrices. The classical approach (1.5) for an inclusion of $\mathbb{PR}(\mathbf{AB})$ requires $2n^3$ switches of the rounding mode and jeopardizes compiler optimization. Also in Profil [18, 19] there is not much cure for this, see Table 1.1. The standard way to compute the product of two interval matrices in midpoint-radius representation is as follows.

$$\langle mA, rA\rangle \cdot \langle mB, rB\rangle \ \subseteq\ \langle\ mAmB\ ,\ |mA|rB + rA(|mB| + rB)\ \rangle\ . \tag{4.15}$$

For $mA, mB, rA, rB \in \mathbb{R}^{n\times n}$, $rA, rB \geq 0$, without the presence of rounding errors, the right hand side includes but overestimates the power set product $\mathbb{PR}(\mathbf{AB})$. Note that in textbooks often the formula $|mA|rB + rA|mB| + rArB$ for the radius is found which requires an additional matrix multiplication.

For an implementation, the two interval matrices $\mathbf{A} = [A_1, A_2], \mathbf{B} = [B_1, B_2] \in \mathbb{IF}^{n\times n}$ in infimum-supremum representation are first converted into midpoint-radius representation, and as in Algorithms 4.2 and 4.3 an inclusion of the products of the midpoints is necessary.

Algorithm 4.5. *Interval matrix times interval matrix, requires* 4 *point matrix multiplications.*

function $[\mathrm{C}_1, \mathrm{C}_2] = \mathrm{IImul}_4(\boldsymbol{A}, \boldsymbol{B})$

$mA := \triangle\big((A_1 + A_2)/2\big);\ rA := \triangle(mA - A_1)$ % $[A_1, A_2] \subseteq \langle mA, rA\rangle$

$mB := \triangle\big((B_1 + B_2)/2\big);\ rB := \triangle(mB - B_1)$ % $[B_1, B_2] \subseteq \langle mB, rB\rangle$

$rC := \triangle\big(|mA| \cdot rB + rA \cdot (|mB| + rB)\big)$ % $\Gamma := |mA| \cdot rB + rA \cdot (|mB| + rB) \leq rC$

$C_2 := \triangle\big(mA \cdot mB + rC\big)$ % $mA \cdot mB + \Gamma \leq C_2$

$C_1 := \nabla\big(mA \cdot mB - rC\big)$ % $C_1 \leq mA \cdot mB - \Gamma$

The algorithm is correct, also in the presence of underflow, there is no restriction in the dimension n, and it requires 4 matrix multiplications.

The radius rB is used twice in (4.15), and the well-known dependency problem [28, Chapter 4] occurs. This is the major source of overestimation of the inclusion $[C_1, C_2]$ computed by Algorithm 4.5. The amount of overestimation has been estimated in [36] using the relative precision of the input intervals.

Definition 4.6. *An interval* $\langle m\boldsymbol{A}, r\boldsymbol{A}\rangle$ *not containing* 0 *is said to be of relative precision* e, $0 \leq e \in \mathbb{R}$, *if*

$$r\boldsymbol{A} \leq e \cdot |m\boldsymbol{A}|\ .$$

An interval containing 0 *is said to be of relative precision* 1.

Suppose all entries in $\mathbf{A}$ and $\mathbf{B}$ are of relative precision e and f, respectively. Then Proposition 2.7 in [36] shows that the maximum overestimation of the result computed by (4.15) compared to $\mathbb{PR}(\mathbf{AB})$ is

$$\text{at most a factor}\ \ 1 + \frac{ef}{e+f}\ \ \text{in radius}\ .$$

This value is globally bounded by 1.5, and it is only possible for wide input including zero. For example, if all entries in $\mathbf{A}$ and $\mathbf{B}$ are at least of relative precision 0.01 or 1%, then the overestimation in radius is bounded by a factor 1.005. So if a radius of the result is 0.01, say, then it is overestimated to 0.01005.

Additional sources of overestimation are the conversion of the input matrices into midpoint-radius representation and the inevitable presence of rounding errors. Except for very narrow input this is negligible compared to the inherent overestimation just discussed, and a thorough analysis or explicit formulas do not reveal much additional insight. A rough analysis shows that by (2.14) and Corollary 3.9 the rounding errors in the computation of the midpoint are bounded by $\gamma_{2n}|mA||mB| \leq \gamma_{2n+2}|\mathrm{mid}(\mathbf{A})||\mathrm{mid}(\mathbf{B})|$, and that of the radius by $\gamma_p|mA||rB| + \gamma_q|rA|(|mB| + rB)$ for p, q of the order n, so that the size of all rounding errors is bounded by

$$\gamma_{\varphi n}|\mathbf{A}||\mathbf{B}| \tag{4.16}$$

for a small constant φ. Thus rounding errors become only relevant for very narrow input $\mathbf{A}, \mathbf{B}$. In that case, however, the result is narrow as well. Nevertheless also a small overestimation may become significant when the result is used in further computations.

As before, one matrix multiplication may be saved by computing the midpoint $mA \cdot mB$ in rounding to nearest and estimating the error by Theorem 2.1.

ALGORITHM 4.7. *Interval matrix times interval matrix, requires* 3 *point matrix multiplications.*

function $[\mathrm{C}_1, \mathrm{C}_2] = \mathrm{IImul}_3(\boldsymbol{A}, \boldsymbol{B})$

$mA := \triangle\big((A_1 + A_2)/2\big);\ rA := \triangle(mA - A_1)$	% $[A_1, A_2] \subseteq \langle mA, rA\rangle$
$mB := \triangle\big((B_1 + B_2)/2\big);\ rB := \triangle(mB - B_1)$	% $[B_1, B_2] \subseteq \langle mB, rB\rangle$
$mC := \square\big(mA \cdot mB\big)$	% *floating-point approximation*
$rB' := \triangle\big((n+2)\mathtt{eps}\lvert mB\rvert + rB\big)$	% *includes error of* $\square(mA \cdot mB)$
$rC := \triangle\big(\lvert mA\rvert \cdot rB' + \mathtt{realmin} + rA \cdot (\lvert mB\rvert + rB)\big)$	% $\Gamma := \lvert mA\rvert \cdot rB + rA \cdot (\lvert mB\rvert + rB) \leq rC$
$C_2 := \triangle\big(mC + rC\big)$	% $mA \cdot mB + \Gamma \leq C_2$
$C_1 := \nabla\big(mC - rC\big)$	% $C_1 \leq mA \cdot mB - \Gamma$

The algorithm is correct, also in the presence of underflow, provided $2(n+2)\mathtt{eps} \leq 1$, and it requires 3 matrix multiplications. The proof of validity follows as in (4.13). A rough analysis shows that all rounding errors together are of the same order $\gamma_{\varphi' n}|\mathbf{A}||\mathbf{B}|$ as in (4.16) with a slightly increased φ'.

Recently Hong Diep Nguyen and Nathalie Revol [29] suggested an alternative diminishing overestimation. For interval matrices $A = \langle mA, rA\rangle, B = \langle mB, rB\rangle$, $mA, mB, rA, rB \in \mathbb{R}^{n\times n}$, $rA, rB \geq 0$, define (using Matlab notation)

$$\rho A := \mathrm{sign}(mA) .* \min(|mA|, rA) \qquad \text{and} \qquad \rho B := \mathrm{sign}(mB) .* \min(|mB|, rB)\ . \tag{4.17}$$

Then, without the presence of rounding errors,

$$\mathbb{PR}\big(\mathbf{AB}\big) \ \subseteq\ \langle\ mAmB + \rho A\rho B\ ,\ |mA|rB + rA(|mB| + rB) - |\rho A||\rho B|\ \rangle\ . \tag{4.18}$$

This can be seen using the fact that for *scalar* intervals $\mathbf{a}, \mathbf{b}$ the true product $\mathbf{c} = \langle m\mathbf{c}, r\mathbf{c}\rangle$ satisfies [28, Proposition 1.6.5]

$$\begin{aligned} m\mathbf{c} &:= m\mathbf{a}m\mathbf{b} + \mathrm{sign}(m\mathbf{a}m\mathbf{b}) \min\big(r\mathbf{a}|m\mathbf{b}|, |m\mathbf{a}|r\mathbf{b}, r\mathbf{a}r\mathbf{b}\big) \\ r\mathbf{c} &:= \max\big(r\mathbf{a}(|m\mathbf{b}| + r\mathbf{b}), (|m\mathbf{a}| + r\mathbf{a})r\mathbf{b}, r\mathbf{a}|m\mathbf{b}| + |m\mathbf{a}|r\mathbf{b}\big)\ . \end{aligned} \tag{4.19}$$

Note that (4.19) does not extend directly to interval matrices. Nguyen and Revol show that the maximum overestimation of (4.18) is at most a factor $4 - 2\sqrt{2} \approx 1.18$ in radius, compared to maximally 1.5 for (4.15). They define the following algorithm to compute an inclusion of $\mathbb{PR}\big(\mathbf{AB}\big)$.

ALGORITHM 4.8. *Interval matrix times interval matrix, requires* 7 *point matrix multiplications.*

function $[\mathrm{C}_1, \mathrm{C}_2] = \mathrm{IImul}_7(\boldsymbol{A}, \boldsymbol{B})$

$mA := \triangle\big((A_1 + A_2)/2\big);\ rA := \triangle(mA - A_1)$ % $[A_1, A_2] \subseteq \langle mA, rA\rangle$

$mB := \triangle\big((B_1 + B_2)/2\big);\ rB := \triangle(mB - B_1)$ % $[B_1, B_2] \subseteq \langle mB, rB\rangle$

$\rho A := \mathrm{sign}(mA) .* \min(|mA|, rA)$

$\rho B := \mathrm{sign}(mB) .* \min(|mB|, rB)$ % *quantities according to* (4.17)

$rC := \triangle\big(|mA| \cdot rB + rA \cdot (|mB| + rB) + (-|\rho A|) \cdot |\rho B|\big)$ % *upper bound for radius* Γ *in* (4.18)

$C_2 := \triangle\big(mA \cdot mB + \rho A \cdot \rho B + rC\big)$ % $mA \cdot mB + \rho A \cdot \rho B + \Gamma \le C_2$

$C_1 := \nabla\big(mA \cdot mB + \rho A \cdot \rho B - rC\big)$ % $C_1 \le mA \cdot mB + \rho A \cdot \rho B - \Gamma$

The algorithm is correct, also in the presence of underflow, there is no restriction on the dimension n, and it requires 7 matrix multiplications.

Again there are three sources of overestimation when implementing (4.18): The inherent overestimation of at most a factor $4 - 2\sqrt{2} \approx 1.18$ in radius, the conversion into midpoint-radius representation and the inevitable rounding errors. A rough analysis uses $|\rho A| = \min(|mA|, rA)$ and $|\rho B| = \min(|mB|, rB)$, (2.14) and Corollary 3.9 to see that the errors in the computation of the midpoint are bounded by

$$\gamma_p(|mA||mB| + |\rho A||\rho B|) \le \gamma_p(|mA| + rA)(|mB| + rB) \le \gamma_q(|\mathbf{A}| + |\mathbf{B}|) \ ,$$

and the rounding errors in the radius are bounded similarly by $\gamma_r(|\mathbf{A}|+|\mathbf{B}|)$ for p, q, r of the order n. So again the total overestimation due to midpoint-radius conversion and rounding errors is bounded by $\gamma_{\psi n}|\mathbf{A}||\mathbf{B}|$ as in (4.16) for small ψ.

In order to develop a faster algorithm, we first rewrite the radius in (4.18) into

$$\begin{aligned} \Gamma &:= |mA|rB + rA(|mB| + rB) - |\rho A||\rho B| \\ &= (|mA| + rA)(|mB| + rB) - \big(|mA||mB| + |\rho A||\rho B|\big) =: \Gamma_1 - \Gamma_2 \ . \end{aligned} \tag{4.20}$$

Note that without the presence of rounding errors, the radius in (4.18) is equal to Γ. From a numerical point of view, a drawback is that for narrow interval matrices two quantities Γ_1, Γ_2 of similar size are subtracted and numerical cancellation may occur. This is visible in the computational results, see Table 5.6.

A rough analysis reveals that this drawback applies only to very narrow intervals. The rounding errors in the computation of Γ_1 are bounded by $\gamma_p(|mA| + rA)(|mB| + rB) \le \gamma_{p'}|\mathbf{A}||\mathbf{B}|$, and those of Γ_2 by $\gamma_q(|mA||mB| + |\rho A||\rho B|) \le \gamma_{q'}|\mathbf{A}||\mathbf{B}|$, so that again, as in (4.16), the total overestimation due to midpoint-radius conversion and rounding errors is bounded by $\gamma_{\psi' n}|\mathbf{A}||\mathbf{B}|$ for small ψ'. Note, however, that error estimates of type (2.13) or (2.14) are absolute bounds and do not reflect a possible cancellation.

The advantage is that, in contrast to (4.18), formulation (4.20) offers the possibility to save two matrix multiplications: By Theorem 2.1 both the error of the midpoint approximation $\square\big(mAmB + \rho A\rho B\big)$ and of the approximation $\mu := \square\big(|mA||mB| + |\rho A||\rho B|\big)$ of the second part Γ_2 of the radius can be estimated using the same quantity μ. The corresponding algorithm is as follows.

ALGORITHM 4.9. *Interval matrix times interval matrix, requires* 5 *point matrix multiplications.*

function $[\mathrm{C}_1, \mathrm{C}_2] = \mathrm{IImul}_5(\boldsymbol{A}, \boldsymbol{B})$

$mA := \triangle\big((A_1 + A_2)/2\big)$; $rA := \triangle(mA - A_1)$ % $[A_1, A_2] \subseteq \langle mA, rA \rangle$

$mB := \triangle\big((B_1 + B_2)/2\big)$; $rB := \triangle(mB - B_1)$ % $[B_1, B_2] \subseteq \langle mB, rB \rangle$

$\rho A := \mathrm{sign}(mA) .* \min(|mA|, rA)$

$\rho B := \mathrm{sign}(mB) .* \min(|mB|, rB)$ % *quantities according to* (4.17)

$mC := \square\big(mA \cdot mB + \rho A \cdot \rho B\big)$ % *midpoint in rounding to nearest*

$\mu := \square\big(|mA| \cdot |mB| + |\rho A| \cdot |\rho B|\big)$ % *used for* Γ_2 *and error bound*

$\gamma := \triangle\big((2n+2)\mathtt{eps}\,\mathrm{ufp}(\mu) + \mathtt{realmin}\big)$ % *error bound*

$rC := \triangle\big((|mA| + rA) \cdot (|mB| + rB) - \mu + 2\gamma\big)$ % *upper bound for* Γ *including error in* mC

$C_2 := \triangle\big(mC + rC\big)$ % $mA \cdot mB + \rho A \cdot \rho B + \Gamma \le C_2$

$C_1 := \nabla\big(mC - rC\big)$ % $C_1 \le mA \cdot mB + \rho A \cdot \rho B - \Gamma$

The algorithm is correct, also in the presence of underflow, provided $2(2n+2)\mathtt{eps} \le 1$, and it requires 5 matrix multiplications.

For the proof of correctness of the Algorithm 4.9 denote $M := mA\,mB + \rho A \rho B \in \mathbb{R}^{n \times n}$, so that

$$\mathbb{PR}\big(\mathbf{AB}\big) \;\subseteq\; \langle\, M \;,\; \Gamma_1 - \Gamma_2 \,\rangle \;. \tag{4.21}$$

by (4.18) and (4.20). The computation of mC can be interpreted as the multiplication of an $n \times 2n$ and an $2n \times n$ matrix. Moreover, it can be assumed that the order of calculation of mC and μ is the same, so that Theorem 2.1 implies

$$|M - mC| \le \square((2n+2)\mathtt{eps}\,\mathrm{ufp}(\mu) + \mathtt{realmin}) \le \gamma \;.$$

Applying Theorem 2.1 again to the approximation μ of Γ_2 gives

$$|\Gamma_2 - \mu| \le \gamma \;.$$

Thus rounding upwards in the computation of rC implies

$$rC \ge \Gamma_1 - (\mu - \gamma) + \gamma \ge \Gamma_1 - \Gamma_2 + \gamma \;,$$

so that

$$C_2 \ge mC + rC \ge mC + \gamma + \Gamma_1 - \Gamma_2 \ge M + \Gamma_1 - \Gamma_2$$

and

$$C_1 \le mC - rC \le mC - \gamma - (\Gamma_1 - \Gamma_2) \le M - (\Gamma_1 - \Gamma_2) \;.$$

Thus (4.21) shows that $\mathbb{PR}\big(\mathbf{AB}\big) \subseteq [C_1, C_2]$ for the result computed by Algorithm 4.9.

5. Computational results. In this section we show computational results for the presented algorithms. We compare three methods for point times interval matrix and five methods for interval times interval matrix as given in Tables 5.1 and 5.2, respectively. Moreover, the influence to the quality of the inclusion in an application such as the solution of linear systems is discussed. In terms of computing time, algorithms based on BLAS3 matrix operations are the only reasonable way to implement interval matrix multiplication in Matlab (cf. [39, Table 9.3]) because other methods suffer severely from interpretation overhead. Therefore

TABLE 5.1
Methods for point matrix times interval matrix.

method	reference	# matrix multiplications
classical	classical method (1.5)	very slow
FImul_3	Algorithm 4.2	3
FImul_2	Algorithm 4.3	2

TABLE 5.2
Methods for interval matrix times interval matrix.

method	reference	# matrix multiplications
classical	classical method (1.5)	very slow
IImul_4	Algorithm 4.5	4
IImul_7	Algorithm 4.8	7
IImul_3	Algorithm 4.7	3
IImul_5	Algorithm 4.9	5

we give computing times measured in Matlab for the algorithms in Tables 5.1 and 5.2 except for the classical method.

For smaller dimension matrix multiplication is so fast that a reasonable timing is difficult, so the numbers in the first rows of Table 5.3 should be read with care. Having said this, for smaller dimension there is a significant interpretation overhead, for larger dimensions the theoretically best possible ratio is almost achieved. Note that in a C-implementation the measured ratio is also for smaller dimensions closer to the theoretical ratio, similar to the results in Table 1.1.

TABLE 5.3
Ratio of computing time in Matlab for algorithms FImul_2, FImul_3, IImul_3, IImul_4, IImul_5 and IImul_7 for random $n \times n$ matrices of different dimension, one floating-point matrix multiplication normed to 1.

	point x interval matrix		interval x interval matrix			
n	FImul_2	FImul_3	IImul_3	IImul_4	IImul_5	IImul_7
100	6.99	7.41	9.80	10.16	19.47	19.25
200	4.56	5.87	6.82	8.23	13.02	15.27
500	4.27	5.58	6.43	7.44	11.01	13.49
1000	3.74	3.76	4.48	5.61	7.31	9.01
2000	2.25	3.20	3.34	4.30	5.82	7.59
5000	2.03	3.02	3.06	4.01	5.20	7.08

Concerning accuracy we generate for point matrix times interval matrix factors A, $\mathbf{B}$ as follows (in Matlab notation):

```
A = randn(n);              % n x n random matrix
B = midrad(randn(n),e);    % random interval matrix, each entry with fixed radius e
```

The statement `randn(n)` generates an $n \times n$-matrix with pseudo-random entries drawn from a normal distribution with mean zero and standard deviation one. The second factor `B` has entries with random midpoint and constant radius `e`. For `e=1` about 70 % of all entries are intervals with zero in the interior, for `e=0.01` about 0.7 %.

Denote the relative precision (according to Definition 4.6) of an interval matrix $\mathbf{B} \in \mathbb{F}^{n \times n}$ by $rel(\mathbf{B}) \in \mathbb{R}^{n \times n}$. Let $\mathbf{C} \in \mathbb{F}^{n \times n}$ be the result of the matrix product computed with one of the methods listed in Table 5.1.

Then Table 5.4 displays the median and maximum of the entrywise ratio of the relative precision of $\mathbf{C}$ and the narrowest (optimal) interval enclosure $A \cdot \mathbf{B}$ as defined by (1.4). For all kind of data the classical method

TABLE 5.4
Point matrix times interval matrix: median and maximum of the ratios of relative precision to the optimal inclusion for randomly generated matrices (fixed dimension $n = 100$ for both factors).

		classical		FImul_3		FImul_2	
n	e	median	max	median	max	median	max
100	1	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
100	0.01	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
100	10^{-5}	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
100	10^{-10}	1.0000	1.0000	1.0000	1.0000	1.0001	1.0001
100	10^{-14}	1.0447	1.2881	1.0454	1.2712	1.8873	2.1983
100	10^{-15}	1.4279	3.1000	1.4343	3.1000	9.2285	12.6346

and FImul_3 based on midpoint radius arithmetic deliver similar results. This seems natural because without the presence of rounding errors both definitions (1.5) and (4.3) are identical and equal to $A \cdot \mathbf{B}$.

But also the faster version FImul_2 with a priori estimation of the error in the floating-point to nearest midpoint computation computes results of similar quality, at least for wide input intervals. For small width of the second factor, however, it deteriorates.

The largest overestimation with the factor 12.6346 should not be overvalued because the radius of the second factor is small. The actual components responsible are

$$[\underline{12.84259857639}101, \underline{12.84259857639}336] \quad \text{and} \quad [\underline{12.84259857639}208, \underline{12.84259857639}227]$$

with a relative precision of $9.1 \cdot 10^{-14}$ and $7.2 \cdot 10^{-15}$, respectively. So in either case the inclusion is tight. The

TABLE 5.5
Point matrix times interval matrix for badly scaled matrices ((4.12) with $f = 5$): median and maximum of the ratios of relative precision to the optimal inclusion (fixed dimension $n = 100$ for both factors).

		classical		FImul_3		FImul_2	
n	e	median	max	median	max	median	max
100	10^{-10}	1.0000	1.0001	1.0000	1.0001	1.0001	1.0001
100	10^{-14}	1.2875	2.1087	1.2985	2.1087	2.1111	2.1569
100	10^{-15}	3.4286	11.2000	3.4815	11.2000	10.3000	12.4000

test was repeated with ill-conditioned input data as those for Table 4.1, but not much difference occurred except that the values for the maximum of the ratios decreased slightly. For badly scaled input data generated by (4.12) with $f = 5$ the results are identical for $e \geq 10^{-10}$, but for smaller values the pictures changes as shown in Table 5.5. We show the results for $f = 5$, for $f = 30$ they are similar. Now the median of overestimation grows both for the classical method as well as for FImul_3, and the worst case for all three algorithms is not too far apart.

Next we show results for interval matrix times interval matrix. First both matrices are generated by `midrad(randn(n),e)`, using the same fixed radius `e` for all entries of both factors. Again results on the ratios of the relative precision to the optimal inclusion are displayed. As can be seen in Table 5.6 there is not much difference between the classical, the midpoint-radius approach IImul_4 and the approach IImul_7 by Nguyen and Revol. As expected, the algorithms IImul_3 and IImul_5 based on a priori floating-point error estimates overestimate the true result, in particular for small width of the input matrices.

TABLE 5.6

Interval matrix times interval matrix: median and maximum of the ratios of relative precision to the optimal inclusion for randomly generated matrices (fixed dimension $n = 100$ for both factors).

	classical		IImul_4		IImul_7		IImul_3		IImul_5	
e	median	max	median	max	median	max	median	max	median	max
100	1.0000	1.0000	1.0046	1.0061	1.0046	1.0061	1.6422	1.7622	1.0046	1.0061
10	1.0000	1.0000	1.0420	1.0559	1.0363	1.0474	6.6361	7.6766	1.0363	1.0474
1	1.0000	1.0000	1.2007	1.2574	1.0188	1.0325	30.0113	35.2913	1.0188	1.0325
0.01	1.0000	1.0000	1.0062	1.0078	1.0000	1.0000	40.6290	48.7160	1.0000	1.0000
10^{-5}	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	40.0357	48.3276	1.0000	1.0000
10^{-10}	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	40.2990	48.1929	1.0001	1.0001
10^{-14}	1.0226	1.1264	1.0231	1.1226	1.0234	1.1226	75.1070	84.1234	1.6547	2.1914
10^{-15}	1.2097	2.0444	1.2129	2.0667	1.2158	2.0667	366.3551	421.7162	6.9774	11.9677

A possibly catastrophic cancellation in a matrix product, which may be hazardous in floating-point computations, is not possible if both factors are interval matrices. Nevertheless the test was repeated for badly scaled input data, generated by (4.12) with $f = 5$. The results are shown in Table 5.7 for $f = 5$, for $f = 30$ they are similar. Again there is not much difference between the classical, the midpoint-radius approach IImul_4 and the approach IImul_7 by Nguyen and Revol. But now algorithm IImul_3 based on a priori floating-point error estimates is much better for small widths of the input matrices, not too far from the first three algorithms.

Despite the fact that the classical method is prohibitively slow, one should also remember that, due to rounding errors, *all* methods overestimate the narrowest inclusion $\mathbf{A} \cdot \mathbf{B}$.

TABLE 5.7

Interval matrix times interval matrix for badly scaled matrices ((4.12) with $f = 5$): median and maximum of the ratios of relative precision to the optimal inclusion (fixed dimension $n = 100$ for both factors).

	classical		IImul_4		IImul_7		IImul_3		IImul_5	
e	median	max	median	max	median	max	median	max	median	max
100	1.0000	1.0000	1.0099	1.0099	1.0098	1.0098	1.0099	1.0099	1.0098	1.0098
10	1.0000	1.0000	1.0909	1.0909	1.0818	1.0818	1.0909	1.0909	1.0818	1.0818
1	1.0000	1.0000	1.5000	1.5000	1.0000	1.0000	1.5000	1.5000	1.0000	1.0000
0.01	1.0000	1.0000	1.0050	1.0050	1.0000	1.0000	1.0050	1.0050	1.0000	1.0000
10^{-5}	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
10^{-10}	1.0000	1.0001	1.0000	1.0001	1.0000	1.0001	1.0001	1.0001	1.0001	1.0002
10^{-14}	1.1432	1.5426	1.1523	1.5484	1.1549	1.5591	1.5591	1.5773	1.9888	2.7128
10^{-15}	2.1333	5.5455	2.1579	5.5455	2.1818	5.6364	5.7059	6.2500	9.3571	15.5455

One may draw the conclusion that, for example, Algorithm IImul_3 with an overestimation up to more than a factor 400 in radius is of not much value. However, this means to complain on a high level. The worst case with an overestimation of a factor 421.7 in radius in Table 5.6 are the intervals

$$[\underline{-1.13508640785746}, \underline{-1.13508640768590}] \quad \text{and} \quad [\underline{-1.13508640777189}, \underline{-1.13508640777147}] \ ,$$

the inclusion by Algorithm IImul_3 and the optimal inclusion, respectively. The relative precision is

$$7.56 \cdot 10^{-11} \quad \text{and} \quad 1.79 \cdot 10^{-13} \ ,$$

respectively, so both inclusions guarantee at least 10 correct figures. Which algorithm should be used depends on the need of such high accuracy, and on whether the results are used in further computations.

To shed additional light on this, we consider the application of interval arithmetic in so-called *verification methods*, namely the computation of an inclusion of the solution of a numerical problem. Such numerical problems cover systems of linear and nonlinear equations, eigenvalue problems, ordinary and partial differential equations etc. For those problems, bounds for a solution are computed including a proof of solvability of the problem and correctness of the bounds. For details, see [28, 39].

Many nonlinear problems eventually lead to the solution of a system of linear equations, for example nonlinear systems [39, Chapter 13], ordinary differential equations (cf. [39, Chapter 15]), or partial differential equations [24], [26], [44], [45], [46]), [39, Chapter 16]. These linear systems often consist of an interval matrix $\mathbf{A}$ and interval right hand side $\mathbf{b}$. In this case all solutions of linear systems $Ax = b$ with $A \in \mathbf{A}$ and $b \in \mathbf{b}$ are included. This is one of the classical applications of interval matrix multiplication.

The following, often used [1, 2, 17] verification algorithm for linear systems is taken from [35]. Following it is presented in executable INTLAB code [39, Chapter 10]. The type casts in the computation of `C` and `Z` ensure that interval operations are used for point input data `A` and/or `b`.

```
function XX = VerifyLinSys(A,b)
  XX = NaN;                          % initialization
  R = inv(mid(A));                   % approximate inverse
  xs = R*mid(b);                     % approximate solution
  xs = xs + R*(mid(b)-mid(A)*xs);  % residual iteration for backward stability
  C = eye(dim(A))-R*intval(A);     % iteration matrix
  Z = R*(b-A*intval(xs));
  X = Z; iter = 0;
  while iter<15                      % interval iteration with epsilon-inflation
    iter = iter+1;
    Y = X*infsup(0.9,1.1) + 1e-20*infsup(-1,1);
    X = Z + C*Y;                     % interval iteration
    if all(in0(X,Y)), XX = xs + X; return; end
  end
```

For a given (interval) matrix A and (interval) right hand side b, the result is either a valid inclusion together with the proof of solvability, i.e. *all* system matrices are nonsingular, or, the result `NaN` indicates that no inclusion could be computed in the given precision. In the latter case, likely the problem is too ill-conditioned. In any case, no false result is possible. Note that proving non-singularity of all matrices within an interval matrix is an NP-hard problem [34].

Multiplications of a point and an interval quantity (vectors are considered as a matrix with one column) are `R*intval(A)`, `R*res` for `res=b-A*intval(xs)`, and `A*intval(xs)`, and the only multiplication of two interval quantities is `C*Y`.

We now replace all those operations by one of the presented algorithms. The test data is generated as follows.

```
A = gallery('randsvd',n,c) * midrad(1,e);
b = A*ones(n,1);
```

It means the midpoint matrix is randomly generated with condition number `c`, the radius `e` introducing a constant relative error for all components, and the right hand side is computed such that the true solution includes the vector of $1's$. Note that the interval matrix most likely contains a singular matrix if the product of the condition number c and the radius e exceeds one. Therefore $ce < 1$ is a natural boundary for verification.

The first Table 5.8 shows all combinations of algorithms for fixed dimension $n = 100$, condition number

$c = 10^{10}$ and radius $e = 10^{-12}$. For reference we compute an inclusion of the true solution set by more sophisticated means. More precisely, an outer and *inner* inclusion is computed (see [39], Section 10.6), which proves that the data in Table 5.8 is correct. We display the median of the entrywise ratios between the relative precision of the inclusion computed by `VerifyLinSys` and the reference inclusion. For example, a displayed value 1.0057 means that using FImul_2 and IImul_5, the inclusion by `VerifyLinSys` is by about 0.6 % worse than the optimal inclusion.

TABLE 5.8

Ratio of relative precision of the result of `VerifyLinSys` *and reference inclusion for fixed dimension* $n = 100$*, condition number* $c = 10^{10}$ *and radius* $e = 10^{-12}$*.*

	classical	IImul_4	IImul_7	IImul_3	IImul_5
classical	1.0000	1.0000	1.0000	1.0000	1.0000
FImul_3	1.0000	1.0000	1.0000	1.0000	1.0000
FImul_2	1.0057	1.0057	1.0057	1.0057	1.0057

As can be seen there is almost no difference whatever combination of method is used. This is true as long the radii of the input data is about $e = 10^{-12}$ or larger. For smaller radii, the overestimation by a priori estimating the error of floating-point operations in FImul_2 introduces increasing overestimation. As an example, for condition number $c = 10^2$ and radius $e = 10^{-14}$ the results are displayed in Table 5.9.

TABLE 5.9

Ratio of relative precision of the result of `VerifyLinSys` *and reference inclusion for fixed dimension* $n = 100$*, condition number* $c = 10^2$ *and radius* $e = 10^{-14}$*.*

	classical	IImul_4	IImul_7	IImul_3	IImul_5
classical	1.0000	1.0000	1.0000	1.0000	1.0000
FImul_3	1.0000	1.0000	1.0000	1.0000	1.0000
FImul_2	1.4962	1.4962	1.4962	1.4962	1.4962

But again criticizing a "large" ratio 1.4962 is lamenting on a high level. The *maximum* ratio of relative precisions of all possibilities using FImul_2 compared to the reference solution was 1.4987. The corresponding component of the inclusion by `VerifyLinSys` and the reference inclusion is

$$[0.99999999997397, 1.00000000002604] \quad \text{and} \quad [0.99999999998268, 1.00000000001732]$$

with a relative precision of $2.60 \cdot 10^{-11}$ and $1.73 \cdot 10^{-11}$, respectively.

Moreover, we choose badly scaled right hand sides such as

$$\texttt{b} = \texttt{exp}(\texttt{f} * \texttt{randn}(\texttt{n}, 1)); \quad \text{and} \quad \texttt{b} = \texttt{A} * \texttt{exp}(\texttt{f} * \texttt{randn}(\texttt{n}, 1));$$

for some factor `f`. For different condition numbers and a moderate factor `f` = 5, the results are similar to those in Tables 5.8 and 5.9. For a larger factor such as `f` = 15, all ratios in the third rows (using FImul_2) increase to about 2 for `b = exp(f*randn(n,1))`, whereas for `b = A*exp(f*randn(n,1))` they are similar or drop to about 1.2. Summarizing, the qualitative behavior does not change significantly.

Up to now, the only multiplication of interval quantities occurred in `C*Y` requiring only $\mathcal{O}(n^2)$ operations. To show the influence of the different algorithms for interval times interval matrix multiplication, we apply Algorithm `VerifyLinSys` to a matrix right hand side. We choose the identity matrix, so that an inclusion of the inverse of all matrices within the input interval matrix is included. In this case the computation of `C*Y` is a multiplication of two interval matrices, and the effect of using the different versions IImul_k can be studied in better detail. We first perform all point times interval matrix multiplications by the standard

midpoint-radius approach FImul$_3$. The results are displayed in Table 5.10. Recall that $ce < 1$ must be satisfied.

TABLE 5.10

Ratio of relative precision of the result of `VerifyLinSys` *and reference inclusion for matrix inversion, fixed dimension* $n = 100$*, and point matrix times interval matrix always by FImul*$_3$*.*

c	e	classical	IImul$_4$	IImul$_7$	IImul$_3$	IImul$_5$
10^2	10^{-4}	1.0000	1.0000	1.0000	1.0000	1.0000
10^2	10^{-8}	1.0000	1.0000	1.0000	1.0000	1.0000
10^5	10^{-8}	1.0000	1.0000	1.0000	1.0000	1.0000
10^2	10^{-12}	1.0000	1.0000	1.0000	1.0000	1.0000
10^5	10^{-12}	1.0000	1.0000	1.0000	1.0000	1.0000
10^{10}	10^{-12}	1.0000	1.0000	1.0000	1.0000	1.0000
10^2	10^{-14}	1.0001	1.0001	1.0001	1.0001	1.0001
10^5	10^{-14}	1.0000	1.0000	1.0000	1.0000	1.0000
10^{10}	10^{-14}	1.0001	1.0001	1.0001	1.0001	1.0001
10^{13}	10^{-14}	1.0020	1.0020	1.0020	1.0012	1.0020

Obviously there is almost no difference using any algorithm IImul$_\ell$ together with Algorithm IImul$_3$. Finally we display the same table using always a priori estimation of rounding errors by FImul$_2$ rather than FImul$_3$. The results are displayed in Table 5.11. Now the inclusion may become 10 times as wide for narrow input data.

TABLE 5.11

Ratio of relative precision of the result of `VerifyLinSys` *and reference inclusion for matrix inversion, fixed dimension* $n = 100$*, and point matrix times interval matrix always by FImul*$_2$*.*

c	e	classical	IImul$_4$	IImul$_7$	IImul$_3$	IImul$_5$
10^2	10^{-4}	1.0000	1.0000	1.0000	1.0000	1.0000
10^2	10^{-8}	1.0000	1.0000	1.0000	1.0000	1.0000
10^5	10^{-8}	1.0000	1.0000	1.0000	1.0000	1.0000
10^2	10^{-12}	1.0109	1.0109	1.0109	1.0109	1.0109
10^5	10^{-12}	1.0109	1.0109	1.0109	1.0109	1.0109
10^{10}	10^{-12}	1.0112	1.0112	1.0112	1.0112	1.0112
10^2	10^{-14}	2.0209	2.0209	2.0209	2.0209	2.0209
10^5	10^{-14}	2.0195	2.0195	2.0195	2.0195	2.0195
10^{10}	10^{-14}	2.0256	2.0256	2.0256	2.0256	2.0256
10^{13}	10^{-14}	2.7290	2.7293	2.7293	2.7237	2.7293
10^2	10^{-15}	7.7992	7.7992	7.7992	7.7992	7.7992
10^5	10^{-15}	7.7884	7.7884	7.7884	7.7884	7.7884
10^{10}	10^{-15}	7.8261	7.8261	7.8261	7.8261	7.8261
10^{13}	10^{-15}	9.4548	9.4554	9.4554	9.4487	9.4554

But once again a large ratio means that the optimal inclusion is overestimated by this factor. For example, for condition number $c = 10^5$ and radius $e = 10^{-15}$ the worst ratio happens for the inclusion components

$$10^3 \cdot [\underline{-2.32788143660617}, \underline{-2.32788139435358}] \quad \text{and} \quad 10^3 \cdot [\underline{-2.32788141797750}, \underline{-2.32788141297821}]$$

with relative precision $9.08 \cdot 10^{-9}$ and $1.07 \cdot 10^{-9}$, respectively, both guaranteeing at least 8 correct figures.

Therefore, from a practical point of view, there seems not too much difference in choosing any of the proposed methods FImul_k or IImul_ℓ.

We note that from a mathematical point of view, the interval iteration in Algorithm `VerifyLinSys` can only be successful if $|I - RA|$ is convergent (rather than $I - RA$ as in point iterations), i.e. has a spectral radius less than 1. In a certain way this is even necessary and sufficient [35].

6. Appendix. To prove Theorem 3.8 note that $m = \mathrm{fl}_\Delta(\mu)$ by Lemma 3.2, so that $m \notin \mathbb{U}$ and (2.9) yield $0 \le m - \mu \le 2\mathtt{eps}\,\mathrm{ufp}(\mu)$, the first inequality in (3.16). To establish a contradiction, suppose the second inequality is not true. Then

$$\begin{array}{rcl} 2\mathtt{eps}\big(|\mu| + \varrho\big) < r - \varrho & = & \Delta(m-a) - \varrho \le m - a + 2\mathtt{eps}\,\mathrm{ufp}(m-a) - \varrho \\ & = & m - \mu + 2\mathtt{eps}\,\mathrm{ufp}(m - \mu + \mu - a) \\ & \le & m - \mu + 2\mathtt{eps}\big(|m-\mu| + |\mu - a|\big) \\ & = & (1 + 2\mathtt{eps})(m - \mu) + 2\mathtt{eps}\varrho \\ & \le & 2\mathtt{eps}(1 + 2\mathtt{eps})\,\mathrm{ufp}(\mu) + 2\mathtt{eps}\varrho\ , \end{array} \tag{6.1}$$

where $m - a \notin \mathbb{U}$ because otherwise $r = \Delta(m-a) = m - a \in \mathbb{U}$ was used, and therefore

$$m > \mu + \frac{2\mathtt{eps}}{1 + 2\mathtt{eps}}\,|\mu| \qquad \text{and} \qquad |\mu| < (1 + 2\mathtt{eps})\,\mathrm{ufp}(\mu)\ . \tag{6.2}$$

If $|\mu| = \mathrm{ufp}(\mu)$ then $m = \mu$, and (2.11) implies $r - \varrho = \Delta(\varrho) - \varrho \le 2\mathtt{eps}\,\mathrm{ufp}(\varrho)$ because $\varrho = m - a \notin \mathbb{U}$, a contradiction. Hence (2.7) and (6.2) yield

$$\mathrm{ufp}(\mu) < |\mu| < (1 + 2\mathtt{eps})\,\mathrm{ufp}(\mu)\ , \tag{6.3}$$

i.e. $|\mu|$ is strictly between $\mathrm{ufp}(\mu)$ and one of its floating-point neighbors. If $\mathrm{ufp}(\mu) \le \mathtt{realmin}/2$, then $m = \mathrm{fl}_\Delta(\mu) \in \mathbb{U}$, thus $\mathrm{ufp}(\mu) \ge \mathtt{realmin}$, and the predecessor and successor of $\mathrm{ufp}(\mu)$ are

$$\mathrm{pred}\big(\mathrm{ufp}(\mu)\big) = (1 - \mathtt{eps})\,\mathrm{ufp}(\mu) \qquad \text{and} \qquad \mathrm{succ}\big(\mathrm{ufp}(\mu)\big) = (1 + 2\mathtt{eps})\,\mathrm{ufp}(\mu)\ . \tag{6.4}$$

Suppose $\mu < 0$. Then, using (6.3) and (6.2),

$$-\mathrm{ufp}(\mu) = \mathrm{fl}_\Delta(\mu) = m > \mu - \frac{2\mathtt{eps}}{1 + 2\mathtt{eps}}\,\mu = \frac{-1}{1 + 2\mathtt{eps}}\,|\mu|\ ,$$

so that $|\mu| > (1 + 2\mathtt{eps})\mathrm{ufp}(\mu)$, contradicting (6.3). It follows $\mu \ge 0$, and $m = \mathrm{fl}_\Delta(\mu) = (1 + 2\mathtt{eps})\mathrm{ufp}(\mu)$ by (6.3) and (6.4). Furthermore, (6.3) and (6.2) yield

$$\mathrm{ufp}(\mu) < \mu < \frac{1 + 2\mathtt{eps}}{1 + 4\mathtt{eps}}\,m < (1 + 4\mathtt{eps}^2)\,\mathrm{ufp}(\mu)\ . \tag{6.5}$$

Suppose $\mathrm{ufp}(\mu) = b/2$. Then $\mathrm{ufp}(\mu) < \mu = \frac{a}{2} + \mathrm{ufp}(\mu)$ together with (6.5) yields $0 < \frac{a}{2} < 4\mathtt{eps}^2\,\frac{b}{2}$ and $\varrho = \frac{b}{2} - \frac{a}{2} > (1 - 4\mathtt{eps}^2)\,\frac{b}{2}$, so that $\mathtt{eps} \le \frac{1}{4}$ and $r = \Delta(m-a) \le m = (1 + 2\mathtt{eps})\,\frac{b}{2}$ give

$$r - \varrho < (2\mathtt{eps} + 4\mathtt{eps}^2)\,\frac{b}{2} < 2\mathtt{eps}\,\mu + \frac{4\mathtt{eps}^2}{1 - 4\mathtt{eps}^2}\,\varrho \le 2\mathtt{eps}(|\mu| + \varrho)\ ,$$

a contradiction to (6.1). Therefore $\mathrm{ufp}(\mu) \ne b/2$, and (6.4) implies

$$\big|\frac{b}{2} - \mathrm{ufp}(\mu)\big| \ge \mathtt{eps}\,\mathrm{ufp}(\mu) \qquad \text{and} \qquad \big|\frac{a}{2}\big| \ge \mathtt{eps}\,\mathrm{ufp}(\mu)\ . \tag{6.6}$$

Suppose $\mathrm{ufp}\big(\frac{a}{2}\big) \ge 2\mathtt{eps}\,\mathrm{ufp}(\mu)$. Then $\frac{a}{2} \in 2\mathtt{eps}\,\mathrm{ufp}\big(\frac{a}{2}\big)\,\mathbb{Z} \subseteq 4\mathtt{eps}^2\mathrm{ufp}(\mu)\,\mathbb{Z}$ by (2.6). If $\frac{b}{2} \ge \mathrm{ufp}(\mu)$, then $\frac{b}{2} \in 2\mathtt{eps}\,\mathrm{ufp}(\mu)\,\mathbb{Z} \subseteq 4\mathtt{eps}^2\mathrm{ufp}(\mu)\,\mathbb{Z}$, and otherwise $\frac{b}{2} < \mathrm{ufp}(\mu)$ and therefore $0 \le a \le b$ give $\frac{b}{2} \in 2\mathtt{eps}\,\mathrm{ufp}(\frac{a}{2})\,\mathbb{Z} \subseteq 4\mathtt{eps}^2\mathrm{ufp}(\mu)\,\mathbb{Z}$. In any case $\mu = \frac{a}{2} + \frac{b}{2} \in 4\mathtt{eps}^2\mathrm{ufp}(\mu)\,\mathbb{Z}$, contradicting (6.5). If $\mathrm{ufp}\big(\frac{a}{2}\big) \le \frac{1}{2}\mathtt{eps}\,\mathrm{ufp}(\mu)$, then $\big|\frac{a}{2}\big| < \mathtt{eps}\,\mathrm{ufp}(\mu)$, a contradiction to (6.6).

It remains the case $\mathrm{ufp}\big(\frac{a}{2}\big) = \mathtt{eps}\,\mathrm{ufp}(\mu)$. This means $\frac{a}{2} \in 2\mathtt{eps}\,\mathrm{ufp}\big(\frac{a}{2}\big)\,\mathbb{Z} = 2\mathtt{eps}^2\mathrm{ufp}(\mu)\,\mathbb{Z}$, so that (6.5) implies $\mu = (1 + 2\mathtt{eps}^2)\mathrm{ufp}(\mu)$. Moreover

$$\mathtt{eps}\,\mathrm{ufp}(\mu) \le \big|\frac{a}{2}\big| < 2\mathtt{eps}\,\mathrm{ufp}(\mu) \tag{6.7}$$

by (2.7), and (6.5) and (6.7) yield

$$\big|\frac{b}{2} - \mathrm{ufp}(\mu)\big| = \big|\mu - \mathrm{ufp}(\mu) + \frac{a}{2}\big| < (2\mathtt{eps}^2 + 2\mathtt{eps})\mathrm{ufp}(\mu)\ .$$

In view of (6.4) this leaves the possibilities $\frac{b}{2} \in \{1 - 2\mathtt{eps}, 1 - \mathtt{eps}, 1 + 2\mathtt{eps}\} \cdot \mathrm{ufp}(\mu)$, where the first implies $\frac{a}{2} = \mu - \frac{b}{2} = \big(2\mathtt{eps}^2 + 2\mathtt{eps}\big)\mathrm{ufp}(\mu)$ contradicting (6.7). Thus only the two floating-point neighbors of $\mathrm{ufp}(\mu)$ are left for $\frac{b}{2}$, and a short computation shows $r - \varrho \le 2\mathtt{eps}(|\mu| + \varrho)$ in both cases. □

7. Conclusion. Several algorithms for interval matrix multiplication have been presented. The algorithms based on midpoint-radius representation are of similar quality as the classical interval matrix multiplication. However, the latter is very slow due to numerous switches of the rounding mode and lack of compiler optimization. Moreover, the former allow the use of fast BLAS3 routines, sequential or parallel.

There are improvements in performance based on a priori estimation of the error of certain floating-point matrix products. They reduce the computing time for point times interval matrix from 3 to 2, and the product of two interval matrices from 4 to 3 matrix multiplications. Although we develop and analyze an optimal conversion between infimum-supremum and midpoint-radius representation, we prefer a slightly weaker but faster method.

Comparing algorithms for interval matrix multiplication directly seems to indicate that those with fewer matrix multiplications are prone to overestimation, in particular for narrow input intervals. In a practical application such as computing error bounds for the solution of a linear system, however, the difference is not too large and often negligible.

Acknowledgement. My thanks to the anonymous referees. I am indebted in particular to one referee, who gave very detailed and helpful advice changing the paper significantly.

REFERENCES

[1] *ACRITH: IBM High-Accuracy Arithmetic Subroutine Library.* IBM Deutschland GmbH, Schönaicher Strasse 220, D-71032 Böblingen, 1986. 3rd edition.

[2] *ARITHMOS, Benutzerhandbuch*, Siemens AG, Bibl.-Nr. U 2900-I-Z87-1 edition, 1986.

[3] J.J. Dongarra, J.J. Du Croz, I.S. Duff, and S.J. Hammarling. A set of level 3 Basic Linear Algebra Subprograms. *ACM Trans. Math. Software*, 16:1–17, 1990.

[4] G.E. Forsythe. Pitfalls in computation, or why a math book isn't enough. *Am. Math. Mon. 77*, pages 931–956, 1970.

[5] A. Frommer. Proving Conjectures by Use of Interval Arithmetic. In U. Kulisch et al., editor, *Perspectives on enclosure methods. SCAN 2000, GAMM-IMACS international symposium on scientific computing, computer arithmetic and validated numerics, Univ. Karlsruhe, Germany, September 19-22, 2000*, Wien, 2001. Springer.

[6] I. Gargantini and P. Henrici. Circular Arithmetic and the Determination of Polynomial Zeros. *Numer. Math.*, 18:305–320, 1972.

[7] D. Goldberg. What Every Computer Scientist Should Know About Floating-Point Arithmetic. *ACM Computaing Surveys*, 23(1):5–47, 1991.

[8] K. Goto and R. Van De Geijn. High-performance implementation of the level-3 blas. *ACM Trans. Math. Softw.*, 35:1–14, 2008.

[9] F. Goualard. How do you compute the midpoint of an interval? Technical report, CNRS, LINA, UMR 6241, Université de Nantes, 2011.

[10] J.R. Hauser. Handling floating-point exceptions in numeric programs. *ACM Trans. Program. Lang. Syst.*, 18(2):139–174, 1996.

[11] N. J. Higham. *Accuracy and stability of numerical algorithms.* SIAM Publications, Philadelphia, 2nd edition, 2002.

[12] *ANSI/IEEE 754-1985: IEEE Standard for Binary Floating-Point Arithmetic.* New York, 1985.

[13] *ANSI/IEEE 754-2008: IEEE Standard for Floating-Point Arithmetic*. New York, 2008.

[14] C. Jacobi, H.J. Oh, K.D. Tran, S.R. Cottier, B.W. Michael, H. Nishikawa, Y. Totsuka, T. Namatame, and N. Yano. The vector floating-point unit in a synergistic processor element of a cell processor. In *ARITH 2005: Proceedings of the 17th IEEE Symposium on Computer Arithmetic*, pages 59–67, Washington, 2005.

[15] Christian Jacobi, Hwa-Joon Oh, Kevin D. Tran, Scott R. Cottier, Brad W. Michael, Hiroo Nishikawa, Yonetaro Totsuka, Tatsuya Namatame, and Naoka Yano. The vector floating-point unit in a synergistic processor element of a Cell processor. In *ARITH '05: Proceedings of the 17th IEEE Symposium on Computer Arithmetic*, pages 59–67, Washington, DC, USA, 2005. IEEE Computer Society.

[16] R.B. Kearfott, M. Dawande, K. Du, and C. Hu. Intlib: A portable Fortran-77 elementary function library. *Interval Comput.*, 3(5):96–105, 1992.

[17] R. Klatte, U. Kulisch, A. Wiethoff, C. Lawo, and M. Rauch. *C-XSC: A C++ Class Library for Extended Scientific Computing*. Springer, Berlin, 1993.

[18] O. Knüppel. PROFIL / BIAS — A Fast Interval Library. *Computing*, 53:277–287, 1994.

[19] O. Knüppel. PROFIL/BIAS and extensions, Version 2.0. Technical report, Inst. f. Informatik III, Technische Universität Hamburg-Harburg, 1998.

[20] M. Malcolm. On accurate floating-point summation. *Comm. ACM*, 14(11):731–736, 1971.

[21] J. Markoff. Writing the Fastest Code, by Hand, for Fun: A Human Computer Keeps Speeding Up Chips . New York Times, November 28, 2005.

[22] R.E. Moore. *Interval Arithmetic and Automatic Error Analysis in Digital Computing*. Dissertation, Stanford University, 1963.

[23] J.M. Muller, N. Brisebarre, F. de Dinechin, C.P. Jeannerod, V. Lefèvre, G. Melquiond, N. Revol, D. Stehlé, and S. Torres. *Handbook of Floating-Point Arithmetic*. Birkhäuser Boston, 2010.

[24] M. T. Nakao. Solving nonlinear elliptic problems with result verification using an H^{-1}type residual iteration. *Computing*, pages 161–173, 1993.

[25] M.T. Nakao. A numerical approach to the proof of existence of solutions for elliptic problems. *Japan J. Appl. Math.*, 5(2):313–332, 1988.

[26] M.T. Nakao, K. Hashimoto, and Y. Watanabe. A numerical method to verify the invertibility of linear elliptic operators with applications to nonlinear problems. *Computing*, 75:1–14, 2005.

[27] A. Neumaier. Vienna proposal for interval arithmetic. http://www.mat.univie.ac.at/ neum/papers.html.

[28] A. Neumaier. *Interval Methods for Systems of Equations*. Encyclopedia of Mathematics and its Applications. Cambridge University Press, 1990.

[29] H.D. Nguyen and N. Revol. Accuracy issues in linear algebra using interval arithmetic. *SCAN conference Lyon*, 2010.

[30] S. Oishi and S.M. Rump. Fast verification of solutions of matrix equations. *Numer. Math.*, 90(4):755–773, 2002.

[31] K. Ozaki, T. Ogita, S. M. Rump, and S. Oishi. Accurate matrix multiplication by using level 3 BLAS operation. In *Proceedings of the 2008 International Symposium on Nonlinear Theory and its Applications, NOLTA'08, Budapest, Hungary*, pages 508–511. IEICE, 2008.

[32] M. Plum. Numerical existence proofs and explicit bounds for solutions of nonlinear elliptic boundary value problems. *Computing*, 49(1):25–44, 1992.

[33] M. Plum. Existence and Multiplicity Proofs for Semilinear Elliptic Boundary Value Problems by Computer Assistance. *DMV Jahresbericht*, 110(1):19–54, 2008.

[34] S. Poljak and J. Rohn. Checking Robust Nonsingularity Is NP-Hard. *Math. of Control, Signals, and Systems 6*, pages 1–9, 1993.

[35] S.M. Rump. *Kleine Fehlerschranken bei Matrixproblemen*. PhD thesis, Universität Karlsruhe, 1980.

[36] S.M. Rump. Fast and parallel interval arithmetic. *BIT Numerical Mathematics*, 39(3):539–560, 1999.

[37] S.M. Rump. INTLAB - INTerval LABoratory. In Tibor Csendes, editor, *Developments in Reliable Computing*, pages 77–104. Kluwer Academic Publishers, Dordrecht, 1999.

[38] S.M. Rump. Inversion of extremely ill-conditioned matrices in floating-point. *Japan J. Indust. Appl. Math. (JJIAM)*, 26:1–29, 2009.

[39] S.M. Rump. Verification methods: Rigorous results using floating-point arithmetic. *Acta Numerica*, 19:287–449, 2010.

[40] S.M. Rump. Error estimation of floating-point summation and dot product. *BIT*, 51(1):201–220, 2012.

[41] S.M. Rump, T. Ogita, and S. Oishi. Accurate floating-point summation part I: Faithful rounding. *SIAM J. Sci. Comput.*, 31(1):189–224, 2008.

[42] P.H. Sterbenz. *Floating-point computation*. Prentice Hall, Englewood Cliffs, NJ, 1974.

[43] T. Sunaga. Geometry of Numerals. Master's thesis, University of Tokyo, February 1956.

[44] A. Takayasu, S. Oishi, and T. Kubo. Guaranteed error estimate for solutions to two-point boundary value problem. In *Proceedings of the International Symposium on Nonlinear Theory and its Applications (NOLTA2009), Sapporo, Japan*, pages 214–217, 2009.

[45] Y. Watanabe. A computer-assisted proof for the Kolmogorov flows of incompressible viscous fluid. *Journal of Computational and Applied Mathematics*, 223:953–966, 2009.

[46] Y. Watanabe, M. Plum, and M.T. Nakao. A computer-assisted instability proof for the Orr-Sommerfeld problem with

Poiseuille flow. *Zeitschrift für Angewandte Mathematik und Mechanik (ZAMM)*, 89(1):5–18, 2009.