ERROR ANALYSIS OF MATRIX MULTIPLICATION WITH NARROW RANGE FLOATING-POINT ARITHMETIC[*]

THEO MARY[†] AND MANTAS MIKAITIS[‡]

Abstract. High-performance computing hardware now supports many different floating-point formats, from 64 bits to only 4 bits. While the effects of reducing precision in numerical linear algebra computations have been extensively studied, some of these low precision formats also possess a very narrow range of representable values, meaning underflow and overflow are very likely. The goal of this article is to analyze the consequences of this narrow range on the accuracy of matrix multiplication. We describe a simple scaling that can prevent overflow while minimizing underflow. We carry out an error analysis to bound the underflow errors and show that they should remain dominated by the rounding errors in most practical scenarios. We also show that this conclusion remains true when multiword arithmetic is used. We perform extensive numerical experiments that confirm that the narrow range of low precision arithmetics should not significantly affect the accuracy of matrix multiplication—provided a suitable scaling is used.

Key words. matrix multiplication, rounding error analysis, floating-point arithmetic, underflow, overflow, scaling, multiword arithmetic, reduced precision, mixed precision, GPUs

AMS subject classifications. 65G50, 65Y04, 65Y10, 68M07

1. Introduction.

The error analysis of computations in floating-point arithmetic traditionally focuses on rounding errors and assumes the absence of underflow and overflow, because the long-standing high precision floating-point arithmetics like binary64 and binary32 of the IEEE 754 standard [11] offer a very wide range of representable values.

Motivated by machine learning applications, there is an emergence of hardware supporting floating-point arithmetics not only with reduced precision, but also with much narrower range than 32- or 64-bit arithmetic. This includes the IEEE 754 binary16 arithmetic [11, Sec. 3.6] and more recently the 4-, 6-, and 8-bit floating-point formats in NVIDIA GPUs [3], standardized by the Open Compute Project standards [14, 16].

This motivates two new developments:

- the incorporation of scaling techniques within standard linear algebra computations in order to prevent overflow and minimize underflow;
- and new error analyses taking into account the effect of underflow, because even with scaling the range of these new formats is so narrow that some amount of underflow is inevitable.

This paper accomplishes this goal for matrix multiplication. We describe a simple diagonal scaling that suffices to prevent overflow and to guarantee that the errors resulting from underflow stay bounded. We carry out an error analysis that shows that these underflow errors should moreover be dominated by the rounding errors in most cases. The narrower range of these emerging low precision arithmetics should therefore not be a cause of major concern in practice. These conclusions extend to the use of multiword arithmetic [7], which we show to reduce both the rounding and underflow errors by a factor of the same order.

The rest of this article is structured as follows. In Section 2, we describe a

[*] Version of January 24, 2025.

[†] Sorbonne Université, CNRS, LIP6, F-75005, Paris, France (theo.mary@lip6.fr).

[‡] School of Computing, University of Leeds, Leeds, LS2 9JT, United Kingdom (m.mikaitis@leeds.ac.uk).

general model of mixed precision matrix multiplication hardware, and provide a list of practical examples that fit into this model. In Section 3, we carry out the rounding error analysis of matrix multiplication with such a hardware model to determine whether underflow errors can become problematic. In Section 4 we extend our analysis to the use of multiword arithmetic. In Section 5 we present extensive numerical experiments that support the conclusions of our analysis. Finally, in Section 6 we provide some concluding remarks.

2. Hardware model and examples.

2.1. The model of matrix multiplication.

MODEL 1 (Mixed precision MMA unit). *We consider a mixed precision matrix multiply-accumulate (MMA) unit with the following specifications to compute a given matrix product $C = AB$. The input matrices A and B must be rounded to a floating-point format, called the input format, which uses t bits of precision and an exponent range $[e_{\min}, e_{\max}]$, leading to:*

- *a unit roundoff equal to $u = 2^{-t}$;*
- *a range of normalized numbers equal to $\pm[f_{\min}, f_{\max}]$, with $f_{\min} = 2^{e_{\min}}$ and $f_{\max} = 2^{e_{\max}}(2-2u)$. If subnormal numbers are available, the interval $[-f_{\min}, f_{\min}]$ contains representable values uniformly spaced by $2uf_{\min}$. The maximum distance between any real number in $[-f_{\min}, f_{\min}]$ and the nearest floating-point number is therefore*

$$g_{\min} = \begin{cases} f_{\min}/2 & \textit{if subnormals are not available} \\ uf_{\min} & \textit{with subnormals (gradual underflow)} \end{cases}. \tag{2.1}$$

The computation of the inner products are performed in a floating-point format, called the accumulation format, which uses $T \ge t$ bits of precision and an exponent range $[E_{\min}, E_{\max}] \supseteq [e_{\min}, e_{\max}]$, leading to:

- *a unit roundoff equal to $U = 2^{-T}$;*
- *a range of normalized numbers equal to $\pm[F_{\min}, F_{\max}]$, with $F_{\min} = 2^{E_{\min}}$ and $F_{\max} = 2^{E_{\max}}(2-2U)$. If subnormal numbers are available, the interval $[-F_{\min}, F_{\min}]$ contains representable values uniformly spaced by $2UF_{\min}$. The maximum distance between any real number in $[-F_{\min}, F_{\min}]$ and the nearest floating-point number is therefore*

$$G_{\min} = \begin{cases} F_{\min}/2 & \textit{if subnormals are not available} \\ UF_{\min} & \textit{with subnormals (gradual underflow)} \end{cases}. \tag{2.2}$$

For both the input and accumulation formats, we use the following model of floating-point arithmetic [6, Equation 2]. Assuming overflow does not occur,

$$\mathrm{fl}(x) = x(1+\delta) + \eta, \quad |\delta| \le u, \quad |\eta| \le g_{\min}, \quad \eta\delta = 0, \tag{2.3}$$

where $\mathrm{fl}(x)$ refers to the operator that rounds $x \in \mathbb{R}$ to the nearest number representable in the input format. Similarly,

$$\mathrm{FL}(x \operatorname{op} y) = (x \operatorname{op} y)(1+\delta) + \eta, \quad |\delta| \le U, \quad |\eta| \le G_{\min}, \quad \eta\delta = 0, \tag{2.4}$$

where $\mathrm{FL}(x \operatorname{op} y)$ refers to the operator rounding $x \operatorname{op} y$ to the accumulation format with $\mathrm{op} \in \{+, -, \times, \div\}$. In the above, the condition $\eta\delta = 0$ refers to the fact that

Table 2.1. *Features of the floating-point formats present in current and recently announced hardware devices. The bottom six formats have narrow ranges whilst the top four have relatively wide ranges. When the formats are used for inputs in Model 1, their parameters are $e_{\min}$, $e_{\max}$, t, $f_{\min}$, $f_{\max}$, and u (listed in the table) and when used in accumulation, the parameters are $E_{\min}$, $E_{\max}$, T, $F_{\min}$, $F_{\max}$, and U, respectively.*

Format	$e_{\min}$	$e_{\max}$	t	$f_{\min}$	$f_{\max}$	u
binary64 (double) [11]	-1022	1023	53	2^{-1022}	$\sim 1.798 \times 10^{308}$	2^{-53}
binary32 (single) [11]	-126	127	24	2^{-126}	$\sim 3.403 \times 10^{38}$	2^{-24}
tf32 (19-bit) [5]	-126	127	11	2^{-126}	$\sim 3.401 \times 10^{38}$	2^{-11}
bfloat16 [12]	-126	127	8	2^{-126}	$\sim 3.389 \times 10^{38}$	2^{-8}
binary16 [11]	-14	15	11	2^{-14}	65504	2^{-11}
fp8-E4M3 [14]	-6	8	4	2^{-6}	448	2^{-4}
fp8-E5M2 [14]	-14	15	3	2^{-14}	57344	2^{-3}
fp6-E2M3 [16]	0	2	4	2^{0}	7.5	2^{-4}
fp6-E3M2 [16]	-2	4	3	2^{-2}	28	2^{-3}
fp4-E2M1 [16]	0	2	2	2^{0}	6	2^{-2}

each error can only be affected by either rounding or underflow, but not both at the same time.

The models (2.3) and (2.4) assume round-to-nearest mode [11, Sec. 4.3]. However, these models continue to hold with directed rounding modes (such as round-to-zero, etc.) by replacing u and $g_{\min}$ (resp. U and $G_{\min}$) with $2u$ and $2g_{\min}$ (resp. $2U$ and $2G_{\min}$). This remark is particularly of interest for NVIDIA tensor cores, which use round-to-nearest for the input format but round-to-zero for the accumulation format.

2.2. Practical examples of MMA hardware. We now provide some practical examples of hardware that satisfy Model 1.

We first list in Table 2.1 the floating-point formats that are available, or were announced to appear, in hardware. The top two formats are the traditional IEEE 754 [11] formats which we consider of relatively wide range and precision. The tf32 and bfloat16 formats that follow them have reduced precision, but have the range almost as wide as the binary32 formats of IEEE 754.

The binary16 format is an IEEE 754 low precision format; we consider it to be a narrow range format since the maximum representable value is 65504—approximately 33 orders of magnitude smaller than $f_{\max}$ of bfloat16.

The Open Compute Project (OCP) 8-bit floating-point specification [14] specifies two 8-bit formats, their encoding, and conversion operations from wider floating-point formats to these two new 8-bit formats. The fp8-E5M2 format is encoded in a 5-bit exponent and a 2-bit fraction, whilst fp8-E4M3 is encoded in a 4-bit exponent and a 3-bit fraction. The values and special values that these formats represent largely follow the rules of IEEE 754, except in a few important cases for fp8-E4M3, as follows. fp8-E4M3 does not represent infinities and represents only two NaN values, a negative and positive. This way fp-E4M3 gains one extra bit pattern for increasing e_{max} to 8 and thus expanding the dynamic range. fp8-E5M2 represents infinities and NaNs as usual, following the rules defined in the IEEE 754 standard, however, no interpretation for the fraction bits of different NaNs is provided. The OCP also provides specification for

TABLE 2.2. *Hardware-accelerated matrix multipliers in the latest data center GPU hardware devices, some of which are prevalent in the TOP500. † According to the instruction set reference manual [1, Table 29] the 8-bit formats on AMD MI300 may not be OCP [14] compliant since they have only one bit pattern for zero and different exponent biases than the OCP formats.*

Device	Input format	Accumulation format
NVIDIA PTX ISA 8.5 [2]	fp8-E5M2	binary32
	fp8-E4M3	binary32
	binary16	binary16
	binary16	binary32
	bfloat16	binary32
	tf32	binary32
AMD MI300 ISA [1, Table 27]	fp8-E5M2†	binary32
	fp8-E4M3†	binary32
	binary16	binary32
	bfloat16	binary32
	tf32 (called xf32 in [1])	binary32

conversion operations from wider floating-point formats to the fp8 formats. It requires saturating and non-saturating conversion modes. Saturating conversion mode returns a maximum value in the target fp8 format when a rounded source value is higher than the maximum representable number; in the non-saturating mode, fp8-E4M3 yields a NaN and fp8-E5M2 an infinity.

Table 2.1 also lists two 6-bit formats and one 4-bit formats, with extremely narrow ranges and precisions introduced in a subsequent OCP standard [16]. These are likely to be available within the forthcoming NVIDIA Blackwell [3] architecture.

We now list in Table 2.2 some examples of hardware that implement MMA units satisfying Model 1, along with the corresponding combinations of input and accumulation formats. This type of MMA unit is commonly found on NVIDIA and AMD GPUs, and are thus very widespread in the modern computing landscape: they appear on more than 100 of the machines in the TOP500[1].

NVIDIA GPUs provide so-called tensor cores units which can use various low precision formats as the input format: binary16, bfloat16, tf32, and since the latest Hopper architecture, both fp8 formats. In almost all cases, the accumulation format is binary32. It is not specified whether this hardware is OCP-compliant.

3. Error analysis of matrix multiplication. Let $A \in \mathbb{R}^{m\times n}$ and $B \in \mathbb{R}^{n\times q}$. Our goal is to compute the product $C = AB$ with the MMA unit of Model 1. Since the coefficients of A and B may not belong to the range of the input format, we must scale them. We compute

$$C = \Lambda^{-1}\Big(\mathrm{fl}(\Lambda A)\,\mathrm{fl}(BM)\Big)M^{-1} \tag{3.1}$$

where Λ and M are nonsingular diagonal matrices with diagonal coefficients λ_i and μ_i, respectively and where $\mathrm{fl}(\cdot)$ denotes the rounding operator (2.3). In order to avoid

[1] https://www.top500.org/lists/top500/list/2024/06/

introducing rounding errors in the scaling, we assume that the scaling factors λ_i and μ_i are all powers of two. Moreover, we also assume the absence of underflows/overflows before the scaling by Λ and M and after the scaling by Λ^{-1} and M^{-1}, that is, we assume that the matrices A, B, and C are all stored in a sufficiently high precision format to be represented exactly.

Let θ be the maximum value that we wish to allow in ΛA and BM. Then, the maximum coefficient on each row of ΛA and each column of BM is in $(\theta/2, \theta]$. There exists an optimal value for θ: as it increases, underflow is minimized, so we must take its largest possible value which prevents overflow from occuring both in the input and in the intermediate computations. Preventing overflow in the input requires $\theta \le f_{\max}$, whereas preventing it in the accumulation requires $\theta \le \sqrt{F_{\max}/n}$. We therefore should set

$$\theta = \min(f_{\max}, \sqrt{F_{\max}/n}). \tag{3.2}$$

3.1. Analysis. Let us now consider the inner product $x^T y$ between a row x of A and a column y of B. The scaling (3.1) leads to

$$s = \lambda^{-1}\mu^{-1}\,\mathrm{fl}(\lambda x)^T\,\mathrm{fl}(\mu y), \tag{3.3}$$

where the scaling factors λ and μ are powers of two satisfying

$$\frac{\theta}{2\|x\|_\infty} < \lambda \le \frac{\theta}{\|x\|_\infty}, \tag{3.4a}$$

$$\frac{\theta}{2\|y\|_\infty} < \mu \le \frac{\theta}{\|y\|_\infty}. \tag{3.4b}$$

Let $\widetilde{x} = \mathrm{fl}(\lambda x)$ and $\widetilde{y} = \mathrm{fl}(\mu y)$; they satisfy

$$\widetilde{x} = \lambda x + \Delta x, \quad |\Delta x| \le u\lambda|x| + g_{\min}, \tag{3.5}$$

$$\widetilde{y} = \mu y + \Delta y, \quad |\Delta y| \le u\mu|y| + g_{\min}, \tag{3.6}$$

where the inequalities hold componentwise: for instance, (3.5) implies $|\Delta x_i| \le u\lambda|x_i| + g_{\min}$ for $i = 1\colon n$. The errors Δx and Δy contain both the conversion error to precision u and the error $g_{\min}$ caused by the underflow of the coefficients of magnitude smaller than $f_{\min}$. Note that each Δx_i may be comprised of only one type of error, conversion or underflow. We obtain

$$s = \lambda^{-1}\mu^{-1}(\lambda x + \Delta x)^T(\mu y + \Delta y) \tag{3.7}$$

$$\quad = x^T y + \varepsilon_1, \tag{3.8}$$

where

$$|\varepsilon_1| \le \lambda^{-1}|\Delta x|^T|y| + \mu^{-1}|x|^T|\Delta y| + \lambda^{-1}\mu^{-1}|\Delta x|^T|\Delta y| \tag{3.9}$$

$$\begin{aligned} &\le (2u+u^2)|x|^T|y| \\ &\qquad + g_{\min}\big((1+u)(\lambda^{-1}e^T|y| + \mu^{-1}|x|^Te) + g_{\min}\lambda^{-1}\mu^{-1}e^Te\big) \end{aligned} \tag{3.10}$$

$$\begin{aligned} &\le (2u+u^2)|x|^T|y| \\ &\qquad + ng_{\min}\big((1+u)(\lambda^{-1}\|y\|_\infty + \mu^{-1}\|x\|_\infty) + g_{\min}\lambda^{-1}\mu^{-1}\big), \end{aligned} \tag{3.11}$$

where in (3.10) e denotes the vector of all ones. Using (3.4), we have $\lambda^{-1} \le 2\theta^{-1}\|x\|_\infty$ and $\mu^{-1} \le 2\theta^{-1}\|y\|_\infty$, which yields

$$\begin{aligned}
|\varepsilon_1| &\le (2u+u^2)|x|^T|y| \\
&\qquad + ng_{\min}\left(4\theta^{-1}(1+u)\|x\|_\infty\|y\|_\infty + 4\theta^{-2}g_{\min}\|x\|_\infty\|y\|_\infty\right) \qquad (3.12)\\
&= (2u+u^2)|x|^T|y| + 4n\theta^{-1}g_{\min}(1+u+\theta^{-1}g_{\min})\|x\|_\infty\|y\|_\infty \qquad (3.13)\\
&= (2u+u^2)|x|^T|y| + 4n\omega(1+u+\omega)\|x\|_\infty\|y\|_\infty \qquad (3.14)
\end{aligned}$$

with

$$\omega = \theta^{-1}g_{\min} = \frac{g_{\min}}{\min\left(f_{\max}, \sqrt{F_{\max}/n}\right)}. \tag{3.15}$$

Note that the appearance of $\|x\|_\infty\|y\|_\infty$, as opposed to $|x|^T|y|$, is unavoidable. To understand why, consider the case where x has one very large coefficient $x_k = \|x\|_\infty$, which corresponds to a very small coefficient y_k. In this case $|x|^T|y|$ can be much smaller than $\|x\|_\infty\|y\|_\infty$ and if $\widetilde{y}$ underflows the error is not bounded by $|x|^T|y|$.

Next, we take into account the errors that occur in the intermediate computations in the accumulation format. We must consider both the rounding errors in precision U and the errors caused by the possible underflow of intermediate coefficients $\mathrm{FL}(\widetilde{x}_k\widetilde{y}_k)$. Assume that r such coefficients underflow, each causing an underflow error bounded by

$$\lambda^{-1}\mu^{-1}G_{\min} \le 4\theta^{-2}G_{\min}\|x\|_\infty\|y\|_\infty. \tag{3.16}$$

The computed $\widehat{s}$ therefore satisfies

$$\widehat{s} = s + \varepsilon_2 \tag{3.17}$$

with

$$\begin{aligned}
|\varepsilon_2| &\le nU|\lambda^{-1}\widetilde{x}|^T|\mu^{-1}\widetilde{y}| + 4r\theta^{-2}G_{\min}\|x\|_\infty\|y\|_\infty \qquad (3.18)\\
&\le nU(|x|^T|y| + |\varepsilon_1|) + 4r\theta^{-2}G_{\min}\|x\|_\infty\|y\|_\infty, \qquad (3.19)
\end{aligned}$$

where the term $nU|x|^T|y|$ accounts for the rounding errors in precision U [13]. Note that we have the bound $r \le n$, because if there are r' multiplications that underflow, there can be at most $n-r'$ underflow errors in the additions. If subnormals are flushed to zero, this is simply because we only have $n-r'$ nonzero terms to sum. With gradual underflow, this is a consequence of Sterbenz lemma [9, Thm. 2.5],[17], by which any addition of two floating-point numbers yielding a subnormal result must be exact.

Combining (3.8) and (3.19), we finally obtain

$$\widehat{s} = x^Ty + \varepsilon_1 + \varepsilon_2 = x^Ty + \varepsilon, \tag{3.20}$$

with

$$\begin{aligned}
|\varepsilon| &\le \big((2u+u^2)(1+nU) + nU\big)|x|^T|y| \\
&\qquad + \big(4n\omega(1+u+\omega)(1+nU) + 4r\theta^{-2}G_{\min}\big)\|x\|_\infty\|y\|_\infty.
\end{aligned} \tag{3.21}$$

Going back to the full matrix product $C = AB$, we have proven that the computed $\widehat{C}$ satisfies

$$\widehat{C} = AB + E \tag{3.22}$$

with

$$\begin{aligned}|e_{ij}| \le \big((2u+u^2)(1+nU)+nU\big)|a_i|^T|b_j| \\ + \big(4n\omega(1+u+\omega)(1+nU)+4r_{ij}\theta^{-2}G_{\min}\big)\|a_i\|_\infty\|b_j\|_\infty,\end{aligned} \tag{3.23}$$

where e_{ij} is the coefficient of E in position (i,j), a_i is the ith row of A, b_j is the jth column of B, and $r_{ij} \le n$ is the number of accumulation underflows in the inner product $a_i^T b_j$. We can weaken this bound in order to bound $\|E\|$ as a function of $\|A\|\|B\|$, for some choice of matrix norm. For example, we have

$$\begin{aligned}\|E\|_\infty \le \Big(\big(2u+u^2+4n^2\omega(1+u+\omega)\big)\big(1+nU\big) \\ + nU + 4n^2\theta^{-2}G_{\min}\Big)\|A\|_\infty\|B\|_\infty.\end{aligned} \tag{3.24}$$

3.2. Summary and discussion. We summarize our analysis in the following theorem.

THEOREM 3.1. *Let $A \in \mathbb{R}^{m\times n}$ and $B \in \mathbb{R}^{n\times q}$ and consider a mixed precision MMA unit as specified in Model 1. Let Λ and M be diagonal matrices whose coefficients are powers of two such that the elements of ΛA and BM are all bounded by*

$$\theta = \min\Big(f_{\max}, \sqrt{F_{\max}/n}\Big).$$

Let $C = AB$ be computed as

$$C = \Lambda^{-1}\Big(\operatorname{fl}(\Lambda A)\operatorname{fl}(BM)\Big)M^{-1}$$

where $\operatorname{fl}(\cdot)$ denotes the rounding operator to the input format and where the inner products in the matrix product $\operatorname{fl}(\Lambda A)\operatorname{fl}(BM)$ are each computed in the accumulation format. Then the computed $\widehat{C}$ satisfies

$$\begin{aligned}\|\widehat{C}-AB\|_\infty \le \Big(\big(2u+u^2+4n^2\theta^{-1}g_{\min}(1+u+\theta^{-1}g_{\min})\big)\big(1+nU\big) \\ + nU + 4n^2\theta^{-2}G_{\min}\Big)\|A\|_\infty\|B\|_\infty.\end{aligned} \tag{3.25}$$

A simplified version of bound (3.25) can be obtained by dropping second-order terms:

$$\|\widehat{C}-AB\|_\infty \lesssim \Big(2u+nU+4n^2\theta^{-1}g_{\min}+4n^2\theta^{-2}G_{\min}\Big)\|A\|_\infty\|B\|_\infty. \tag{3.26}$$

This bound can be compared with the one obtained with a mixed precision MMA unit without considering the range limitations (that is, assuming underflow does not occur). A componentwise bound is given by Blanchard et al. [4, Thm. 3.2],

$$|\widehat{C}-AB| \lesssim (2u+nU)|A||B|, \tag{3.27}$$

which implies the normwise bound

$$\|\widehat{C}-AB\|_\infty \lesssim (2u+nU)\|A\|_\infty\|B\|_\infty. \tag{3.28}$$

Bound (3.26) thus recovers the term $2u+nU$ coming from the rounding errors. In addition, it also accounts for the errors caused by underflows with the terms $4n^2\theta^{-1}g_{\min}$ (input underflows) and $4n^2\theta^{-2}G_{\min}$ (accumulation underflows).

Several comments are in order. In most cases we can expect $\theta \geq 1$, in which case we have $\theta^{-2} G_{\min} \leq \theta^{-1} g_{\min}$ since $U \leq u$ and $F_{\min} \leq f_{\min}$, and therefore the error generated by accumulation underflows is anyway bounded by the underflows in the input. It is possible to have $\theta < 1$ if $n > F_{\max}$, but this is conceivable only for very large matrices and an accumulation format with a narrow range such as binary16. Therefore, in most cases, we can expect accumulation underflows to be insignificant compared with input underflows.

Regarding the latter, if gradual underflow is used, then $g_{\min} = u f_{\min} \ll u$ and thus the input underflows are dominated by the rounding errors. If subnormals are not used, then $g_{\min} = f_{\min}/2$. If $F_{\max}$ is sufficiently large so that $f_{\max} \leq \sqrt{F_{\max}/n}$ and thus $\theta = f_{\max}$, then $\omega = \theta^{-1} f_{\min}/2 = f_{\min}/(2 f_{\max})$ corresponds to twice the inverse width of the range of the input format. For all formats of interest, ω is smaller than the corresponding unit roundoff u and hence we can expect the rounding errors to remain the dominant source of errors, even with subnormals stored as zero. However, since the ω term depends on n^2 whereas the u term does not depend on n, underflows may become problematic for large matrices.

Finally, note that if underflows occur we can only obtain a normwise bound, as opposed to a componentwise one like (3.27). This is due to the appearance of $\|x\|_\infty \|y\|_\infty$ instead of $|x|^T |y|$ in the error analysis, which is unavoidable as mentioned before.

Overall, can the underflows become a cause of concern? There cannot be a definitive answer in general, because it depends on the properties of the input and accumulation formats, as well as the inner dimension n of the matrices. However, in practice, for most cases of interest, we can expect the answer to be no: the underflows should remain dominated by the rounding errors, and thus the range limitations should not significantly affect the accuracy of the computation—provided a suitable scaling is used, naturally. It is indeed worth noting that if the diagonal scaling (3.1) is not used, we are no longer able to obtain informative bounds—in fact, the relative accuracy of the computed result can be arbitrarily bad.

4. Multiword arithmetic. Multiword arithmetic is a popular approach to enhance the accuracy of computations. It consists in representing numbers as the unevaluated sum of low precision floating-point numbers $\sum_{i=0}^{p-1} w^{(i)}$, where each $w^{(i)}$ is a low precision "word", and where p is the number of words. Given a number $w \in \mathbb{R}$, its multiword decomposition can be computed as

$$w^{(i)} = \mathrm{fl}\left(w - \sum_{j=0}^{i-1} w^{(j)}\right), \quad i = 0\colon p-1, \tag{4.1}$$

where $\mathrm{fl}(\cdot)$ rounds to the target low precision format as per (2.4). Denoting by u its unit roundoff, the resulting decomposition achieves an error of order u^p:

$$w + \Delta w = \sum_{i=0}^{p-1} w^{(i)}, \quad |\Delta w| \leq u^p |w|. \tag{4.2}$$

This can be applied elementwise to matrices A and B to obtain multiword decompositions $A \approx \sum_{i=0}^{p-1} A^{(i)}$ and $B \approx \sum_{j=0}^{p-1} B^{(j)}$. The matrix product $C = AB$ can then be evaluated as the sum of the $p(p+1)/2$ products $A^{(i)} B^{(j)}$ of lowest order (those for which $i + j < p$). Fasi et al. [7] carry out error analysis of this approach based on a general MMA model similar to the one used by Blanchard et al. [4] (which thus does not take into account range limitations). They obtain an error bound of order $\max(u^p, U)$; see (4.33).

In the following, we seek to analyze the effect of range limitations on the accuracy of this multiword approach, that is, we extend the analysis of the previous section made for a "single word" product to the case of a general p-word product.

4.1. Analysis. First, note that with the decomposition (4.1), we have $|w^{(i)}| \le u^i(1+u)|w|$, so that the magnitude of the words $w^{(i)}$ quickly decreases as i increases. Without a suitable scaling these words would therefore quickly underflow. In order to reduce the amount of underflow, we therefore employ scaling, as suggested by Ootomo and Yokota [15] in the case $p = 2$, by slightly adapting the decomposition as follows:

$$w^{(i)} = \mathrm{fl}\Big((w - \sum_{j=0}^{i-1} u^j w^{(j)})/u^i\Big), \quad i = 0\colon p-1, \tag{4.3}$$

which yields

$$w + \Delta w = \sum_{i=0}^{p-1} u^i w^{(i)}, \quad |\Delta w| \le u^p |w|. \tag{4.4}$$

The idea is then to apply this decomposition elementwise on the scaled matrices ΛA and BM: we compute

$$A^{(i)} = \mathrm{fl}\Big(\Big(\Lambda A - \sum_{k=0}^{i-1} u^k A^{(k)}\Big)/u^i\Big) \tag{4.5}$$

$$B^{(j)} = \mathrm{fl}\Big(\Big(BM - \sum_{k=0}^{j-1} u^k B^{(k)}\Big)/u^j\Big) \tag{4.6}$$

and we approximate $C = AB$ as

$$C \approx \Lambda^{-1}\Big(\sum_{i+j<p} u^{i+j} A^{(i)} B^{(j)}\Big) M^{-1}. \tag{4.7}$$

Let us begin by bounding the accuracy of the multiword decomposition of A and B. Let us consider any row x of A and any column y of B, and denote $x^{(i)}$ and $y^{(j)}$ the corresponding rows and columns of $A^{(i)}$ and $B^{(j)}$, respectively. We also denote λ the corresponding scaling factor for row x and x_k the kth coefficient of x. Let us prove by induction on ℓ that we have

$$\lambda x_k = \sum_{i=0}^{\ell} u^i x_k^{(i)} + \Delta x_k^{(\ell)}, \quad |\Delta x_k^{(\ell)}| \le \max(u^{\ell+1}|\lambda x_k|, u^\ell g_{\min}). \tag{4.8}$$

For $\ell = 0$, we have $x_k^{(0)} = \mathrm{fl}(\lambda x_k) = \lambda x_k + \Delta x_k^{(0)}$, where $|\Delta x_k^{(0)}| \le u|\lambda x_k|$ if $x_k^{(0)}$ does not underflow, and if it does, $|\Delta x_k^{(0)}| \le g_{\min}$. Now, assuming (4.8) holds for some ℓ, then $x_k^{(\ell+1)} = \mathrm{fl}(\Delta x_k^{(\ell)}/u^{\ell+1})$ satisfies

$$x_k^{(\ell+1)} = \frac{\Delta x_k^{(\ell)}}{u^{\ell+1}} + \varepsilon \tag{4.9}$$

where

$$|\varepsilon| \le u\frac{|\Delta x_k^{(\ell)}|}{u^{\ell+1}} \le \max(u|\lambda x_k|, g_{\min}) \tag{4.10}$$

if $x_k^{(\ell+1)}$ does not underflow and $|\varepsilon| \le g_{\min}$ if it does underflow. In either case we have

$$u^{\ell+1} x_k^{(\ell+1)} = \mathit{\Delta} x_k^{(\ell)} + \mathit{\Delta} x_k^{(\ell+1)} \tag{4.11}$$

with

$$|\mathit{\Delta} x_k^{(\ell+1)}| = |u^{\ell+1}\varepsilon| \le \max(u^{\ell+2}|\lambda x_k|, u^{\ell+1} g_{\min}). \tag{4.12}$$

This concludes the induction proof.

Since (4.8) holds for all coefficients x_k of x, we obtain the vector inequality

$$\lambda x = \sum_{i=0}^{p-1} u^i x^{(i)} + \mathit{\Delta} x^{(p-1)}, \quad |\mathit{\Delta} x^{(p-1)}| \le \max(u^p|\lambda x|, u^{p-1} g_{\min}) \tag{4.13}$$

where the max() is componentwise and where we have taken $\ell = p-1$. Hence

$$x = \sum_{i=0}^{p-1} \lambda^{-1} u^i x^{(i)} + \mathit{\Delta} x, \tag{4.14}$$

where since $\lambda^{-1} \le 2\theta^{-1}\|x\|_\infty$

$$\begin{aligned} |\mathit{\Delta} x| = |\lambda^{-1}\mathit{\Delta} x^{(p-1)}| &\le \max(u^p|x|, \lambda^{-1}u^{p-1} g_{\min}) && (4.15)\\ &\le \max(u^p|x|, 2u^{p-1}\theta^{-1} g_{\min}\|x\|_\infty). && (4.16)\end{aligned}$$

Similarly we have for y

$$y = \sum_{j=0}^{p-1} \mu^{-1} u^j y^{(j)} + \mathit{\Delta} y, \quad |\mathit{\Delta} y| \le \max(u^p|y|, 2u^{p-1}\theta^{-1} g_{\min}\|y\|_\infty). \tag{4.17}$$

Going back to the full matrices A and B, we have therefore proven

$$\begin{aligned} A &= \sum_{i=0}^{p-1} u^i \Lambda^{-1} A^{(i)} + \mathit{\Delta} A, \quad \|\mathit{\Delta} A\|_\infty \le \zeta \|A\|_\infty && (4.18)\\ B &= \sum_{j=0}^{p-1} u^j B^{(j)} M^{-1} + \mathit{\Delta} B, \quad \|\mathit{\Delta} B\|_\infty \le \zeta \|B\|_\infty && (4.19)\end{aligned}$$

with

$$\zeta = \max(u^p, 2nu^{p-1}\theta^{-1} g_{\min}). \tag{4.20}$$

This generalizes the multiword decomposition error bound of order u^p by taking into account possible underflows, which add the error term $2nu^{p-1}\theta^{-1}g_{\min}$.

It now remains to bound the accuracy of the product $C = AB$ approximated with (4.7) and computed in the accumulation format. Denoting $\mathit{\Delta} C$ the errors in the computation of the products $A^{(i)}B^{(j)}$ in the accumulation format, the computed $\widehat{C}$

satisfies

$$
\begin{aligned}
\widehat{C} &= \Lambda^{-1}\Big(\sum_{i+j<p} u^{i+j} A^{(i)} B^{(j)} + \mathit{\Delta} C\Big) M^{-1} && (4.21)\\
&= \Lambda^{-1}\Big(\sum_{i=0}^{p-1}\sum_{j=0}^{p-1} u^{i+j} A^{(i)} B^{(j)} - \sum_{i+j\ge p} u^{i+j} A^{(i)} B^{(j)} + \mathit{\Delta} C\Big) M^{-1} && (4.22)\\
&= \sum_{i=0}^{p-1} u^i \Lambda^{-1} A^{(i)} \sum_{j=0}^{p-1} u^j B^{(j)} M^{-1} + \Lambda^{-1}\Big(\mathit{\Delta} C - \sum_{i+j\ge p} u^{i+j} A^{(i)} B^{(j)}\Big) M^{-1} && (4.23)\\
&= (A - \mathit{\Delta} A)(B - \mathit{\Delta} B) + \Lambda^{-1}\mathit{\Delta} C M^{-1} - \sum_{i+j\ge p} u^{i+j}\Lambda^{-1} A^{(i)} B^{(j)} M^{-1} && (4.24)\\
&= AB + E. && (4.25)
\end{aligned}
$$

Since by construction $|A^{(i)}| \le (1+u)|\Lambda A|$ and $|B^{(j)}| \le (1+u)|BM|$, we have

$$\Big\| \sum_{i+j\ge p} u^{i+j}\Lambda^{-1} A^{(i)} B^{(j)} M^{-1} \Big\|_\infty \lesssim (p-1) u^p \|A\|_\infty \|B\|_\infty. \qquad (4.26)$$

To bound the error $\mathit{\Delta} C$, [7, Theorem 2.1] obtains a relative error bound of order $(n+p^2)U$, to which we must add the accumulation underflows. Proceeding identically to the single word analysis of the previous section, we obtain

$$\|\mathit{\Delta} C\|_\infty \lesssim \Big((n+p^2)U + 4rn\theta^{-2} G_{\min}\Big)\|A\|_\infty \|B\|_\infty, \qquad (4.27)$$

where r bounds the maximum number of possible underflows in the computation of any coefficient of C: for the same reason as in the $p = 1$ case, we have the bound $r \le np(p+1)/2$. In conclusion we have proven

$$
\begin{aligned}
\|E\|_\infty \lesssim \Big(&2\zeta + \zeta^2 + (p-1)u^p\\
&+ (n+p^2)U + 2p(p+1)n^2\theta^{-2} G_{\min}\Big)\|A\|_\infty \|B\|_\infty,
\end{aligned}
\qquad (4.28)
$$

where ζ is defined in (4.20).

4.2. Summary and discussion. We summarize the extension of Theorem 3.1 to multiword arithmetic in the following theorem.

Theorem 4.1. *Let $A \in \mathbb{R}^{m\times n}$ and $B \in \mathbb{R}^{n\times q}$ and consider a mixed precision MMA unit as specified in Model 1. Let Λ and M be diagonal matrices whose coefficients are powers of two such that the elements of ΛA and BM are all bounded by*

$$\theta = \min\Big(f_{\max}, \sqrt{F_{\max}/n}\Big).$$

Let the p-word decompositions $\Lambda A \approx \sum_{i=0}^{p-1} u^i A^{(i)}$ and $BM \approx \sum_{j=0}^{p-1} u^j B^{(j)}$ be computed as

$$A^{(i)} = \mathrm{fl}\Big(\Big(\Lambda A - \sum_{k=0}^{i-1} u^k A^{(k)}\Big)/u^i\Big) \qquad (4.29)$$

$$B^{(j)} = \mathrm{fl}\Big(\Big(BM - \sum_{k=0}^{j-1} u^k B^{(k)}\Big)/u^j\Big) \qquad (4.30)$$

where $\mathrm{fl}(\cdot)$ *denotes the rounding operator to the input format. Let* $C = AB$ *be computed as*

$$C = \Lambda^{-1}\Big(\sum_{i+j<p} u^{i+j} A^{(i)} B^{(j)}\Big) M^{-1}, \tag{4.31}$$

where the inner products in the matrix products $A^{(i)}B^{(j)}$ *are all computed in the accumulation format. Then the computed* $\widehat{C}$ *satisfies*

$$\begin{aligned} \|\widehat{C} - AB\|_\infty \lesssim \Big(&(p+1)u^p + 4nu^{p-1}\theta^{-1}g_{\min} \\ &+ (n+p^2)U + 2p(p+1)n^2\theta^{-2}G_{\min}\Big)\|A\|_\infty\|B\|_\infty. \end{aligned} \tag{4.32}$$

Bound (4.32) is to be compared with the bound obtained by Fasi et al. [7, Theorem 2.1] without taking into account range limitations:

$$\|\widehat{C} - AB\|_\infty \lesssim \Big((p+1)u^p + (n+p^2)U\Big)\|A\|_\infty\|B\|_\infty. \tag{4.33}$$

The comparison shows that we recover the same two terms of order u^p and U, and we have two new terms of order $u^{p-1}\theta^{-1}g_{\min}$ and $\theta^{-2}G_{\min}$ to account for the input and accumulation underflows, respectively.

More importantly, we should compare bound (4.32) with bound (3.26) obtained by Theorem 3.1, to check whether the conclusions of the previous sections continue to hold with multiword arithmetic. The comparison shows that the error terms associated with the input format, $2u$ and $4n\theta^{-1}g_{\min}$ in (3.26), are both reduced with multiword arithmetic by a factor of order u^{p-1}, giving $(p+1)u^p$ and $4nu^{p-1}\theta^{-1}g_{\min}$ in (4.32). Therefore, the main conclusion of Theorem 4.1 is that the use of multiword arithmetic reduces in equal measure the rounding errors and the underflow errors associated with the input format; hence we can expect the latter to remain dominated by the former even when $p > 1$.

If the number of words p is large enough so that $u^p \lesssim U + \theta^{-2}G_{\min}$, the errors associated with the accumulation format will dominate. However, for the same reasons as discussed in the previous section, we can expect the accumulation underflow errors of order $\theta^{-2}G_{\min}$ to be insignificant and remain dominated by the rounding errors of order U.

Overall, we conclude once more that underflows should not be a cause for concern, even when rounding errors are reduced with the use of multiword arithmetic.

5. Numerical experiments.

5.1. Experimental setting. For the numerical experiments we have utilized the CPFloat[2] package [8] that allows for simulating custom-precision and custom-range floating-point formats on a conventional hardware containing IEEE 754 binary64 arithmetic. For this work we have extended the package to be able to simulate formats with exponent limits $e_{\min}$ and $e_{\max}$ that do not have the constraint $e_{\min} = 1 - e_{\max}$ imposed by the IEEE 754 standard. This allowed us to simulate the precision and range of the fp8-E4M3 format of the OFP8 standard [14] with $e_{\min} = -6$ and $e_{\max} = 8$.

The elements of the matrices $A \in \mathbb{R}^{m\times n}$ and $B \in \mathbb{R}^{n\times q}$ are generated as $\pm 10^{\phi}$ where the sign ± is randomly chosen with equal probability and where ϕ is randomly

[2] https://github.com/north-numerical-computing/cpfloat

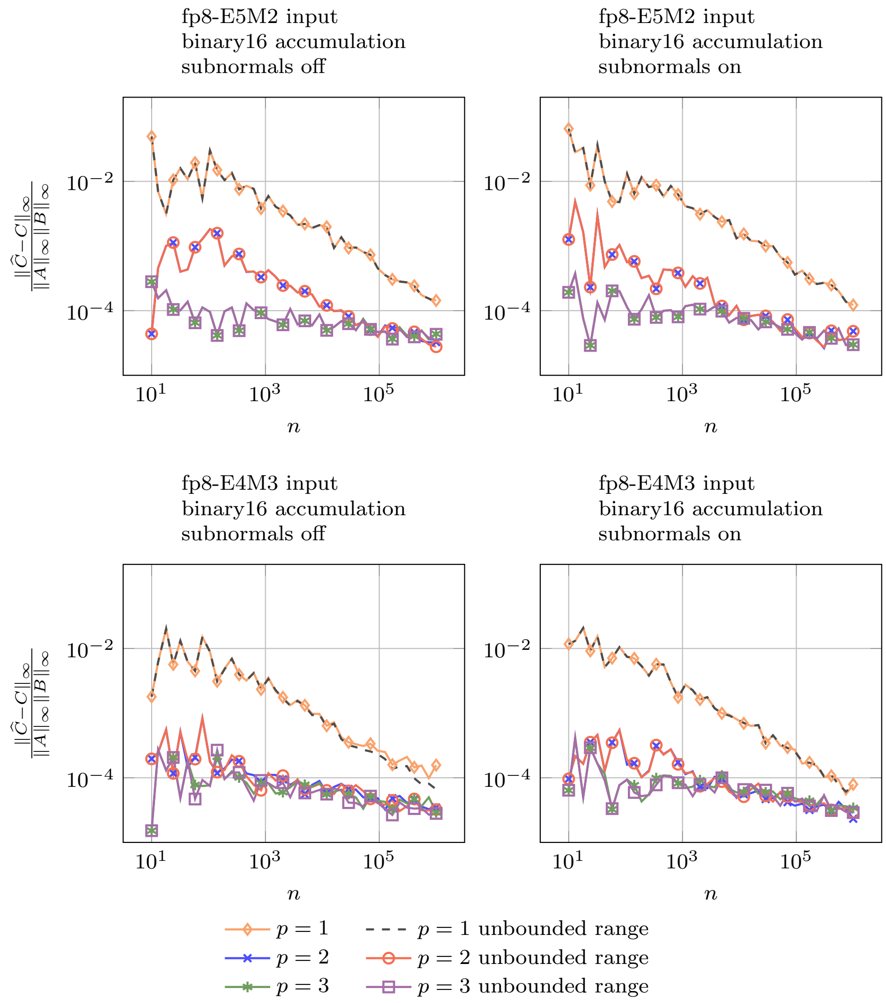


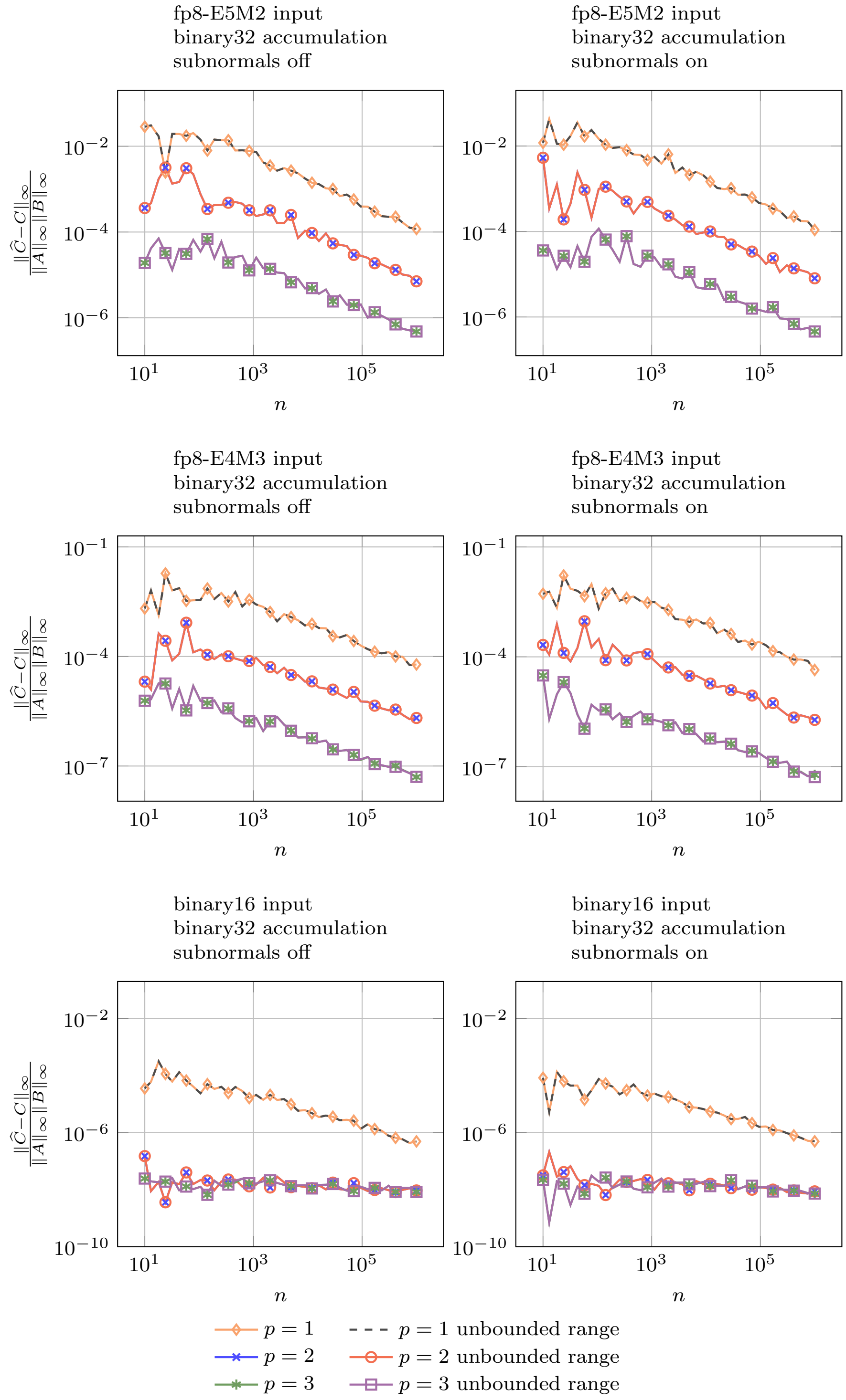
FIG. 5.1. *Normwise relative errors of Model 1 with two 8-bit input formats and the accumulation format binary16, compared with MATLAB's binary64 matrix multiplication. Above each figure the input-accumulation format combination and the use of subnormals are specified.*

sampled from the uniform $[-\ell,\ell]$ distribution using MATLAB's `rand`. This generates matrix elements of magnitudes in the range $[10^{-\ell},10^{\ell}]$. We tested various values for ℓ without observing significant differences in the results. In the following, we use $\ell=10$. For the matrix dimensions, we fix the outer dimensions $m=q=10$ and vary the inner dimension n.

We measure the accuracy of matrix multiplication with the normwise relative error

$$\frac{\|\widehat{C}-C\|_\infty}{\|A\|_\infty\|B\|_\infty}, \tag{5.1}$$

where the "exact" C is computed in binary64 arithmetic.



fp8-E5M2 input
binary32 accumulation
subnormals off

fp8-E5M2 input
binary32 accumulation
subnormals on

fp8-E4M3 input
binary32 accumulation
subnormals off

fp8-E4M3 input
binary32 accumulation
subnormals on

binary16 input
binary32 accumulation
subnormals off

binary16 input
binary32 accumulation
subnormals on

FIG. 5.2. *Normwise relative errors of Model 1 with one 16-bit input format and two 8-bit formats and the accumulation format binary32, compared with MATLAB's binary64 matrix multiplication. Above each figure the input-accumulation format combination and the use of subnormals are specified.*

The data files and code for reproducing the experiments are available[3].

5.2. Experimental results. Figures 5.1 and 5.2 report the normwise relative error (5.1) in various settings. On each row of each figure, we test various combinations of input and accumulation formats: Figure 5.1 uses fp8 (E4M3 or E5M2) for the input and binary16 for the accumulation, whereas Figure 5.2 uses fp8 or binary16 for the input and binary32 for the accumulation. In each figure, the left and right columns plot the results without and with subnormals, respectively. Finally, in each graph, we test various number of words $p \in \{1, 2, 3\}$, corresponding to single-, double-, and triple-word arithmetic as detailed in Section 4.

In each setting, we compare the error when using Model 1 with the error obtained with an unbounded exponent range. The latter is naturally impractical, since it is not implemented in hardware; we simply use it to determine whether the accuracy is limited by the range or by the precision. Indeed, if the two errors overlap then we can conclude that the narrow range has not been a limiting factor, whereas if they diverge it means that the underflows have increased the error.

The results show that in almost all cases the two types of error overlap, which confirms the theoretical conclusions of our error analysis: underflows do not contribute significantly to the overall error in matrix multiplication with narrow range formats. Importantly, this is true both for single-word and multiword arithmetic: for example, in Figure 5.2, using triple-word arithmetic ($p = 3$) with fp8-E4M3 input achieves an accuracy of order at least 10^{-5}, despite its very narrow range.

The only case where the errors with narrow range and unbounded range slightly diverge is in the bottom-left subplot of Figure 5.1. This corresponds to the input format fp8-E4M3 and the accumulation format binary16, with subnormals off. The divergence only occurs for the largest values of n. The issue here is that for large values of n and an accumulation format with relatively narrow range such as binary16, θ in (3.2) is equal to $\sqrt{F_{\max}/n} = \sqrt{65504/n}$. Thus when n exceeds 65504, θ becomes smaller than 1 and as n increases small values of θ will lead to more input underflows. The divergence disappears when using subnormals and/or binary32 accumulation, and is in any case very small.

6. Conclusion.

We have analyzed the accuracy of matrix multiplication when using floating-point arithmetic with a narrow range of representable values. We have used Model 1, a model of mixed precision matrix multiply–accumulate hardware which is satisfied for several practical examples of hardware, notably the NVIDIA GPU tensor cores.

We have proposed in (3.1) a simple scaling for matrix multiplication that suffices to prevent overflow and minimize underflow. Using this scaling and under Model 1, we have obtained in Theorem 3.1 an error bound that takes into account both the rounding errors and underflow errors. The bound suggests that the underflow errors should be dominated by the rounding errors in most practical cases: in particular, this is the case when subnormals and/or high precision accumulation are used. We have proved that these conclusions extend to the use of multiword arithmetic in Theorem 4.1. We have also performed extensive numerical experiments that confirm these conclusions.

Overall, we therefore expect that the narrow range of emerging low precision arithmetics should not be a limiting factor to the accuracy of matrix multiplication. We expect this conclusion to extend to most standard linear algebra computations,

[3] https://github.com/north-numerical-computing/narrow-range-FP-underflow-experiments

provided a suitable scaling can be found—such as the one proposed in [10] for the solution of linear systems.

Acknowledgments. This work was supported by the NumPEx Exa-MA (ANR-22-EXNU-0002), MixHPC (ANR-23-CE46-0005-01), and InterFLOP (ANR-20-CE46-0009) projects of the French National Research Agency (ANR).

REFERENCES

[1] *"AMD Instinct MI300" Instruction Set Architecture. Reference Guide*, Advanced Micro Devices, Inc., Dec. 2023, https://www.amd.com/content/dam/amd/en/documents/instinct-tech-docs/instruction-set-architectures/amd-instinct-mi300-cdna3-instruction-set-architecture.pdf.

[2] *CUDA PTX ISA*, NVIDIA, May 2024, https://docs.nvidia.com/cuda/pdf/ptx_isa_8.5.pdf. Release 8.5.

[3] *NVIDIA Blackwell Architecture Technical Brief*, NVIDIA, Mar. 2024, https://nvdam.widen.net/s/xqt56dflgh/nvidia-blackwell-architecture-technical-brief. V1.0.

[4] P. Blanchard, N. J. Higham, F. Lopez, T. Mary, and S. Pranesh, *Mixed precision block fused multiply-add: Error analysis and application to GPU tensor cores*, SIAM J. Sci. Comput., 42 (2020), pp. C124–C141, https://doi.org/10.1137/19M1289546.

[5] J. Choquette, E. Lee, R. Krashinsky, V. Balan, and B. Khailany, *3.2 the A100 datacenter GPU and Ampere architecture*, in Proceedings of the 2021 IEEE International Solid-State Circuits Conference, San Francisco, CA, USA, Mar. 2021, pp. 48–50, https://doi.org/10.1109/ISSCC42613.2021.9365803.

[6] J. Demmel, *Underflow and the reliability of numerical software*, SIAM Journal on Scientific and Statistical Computing, 5 (1984), pp. 887–919, https://doi.org/10.1137/0905062.

[7] M. Fasi, N. J. Higham, F. Lopez, T. Mary, and M. Mikaitis, *Matrix multiplication in multiword arithmetic: Error analysis and application to GPU tensor cores*, SIAM J. Sci. Comput., 45 (2023), pp. C1–C19, https://doi.org/10.1137/21m1465032.

[8] M. Fasi and M. Mikaitis, *CPFloat: A C library for simulating low-precision arithmetic*, ACM Trans. Math. Softw., 49 (2023), https://doi.org/10.1145/3585515.

[9] N. J. Higham, *Accuracy and Stability of Numerical Algorithms*, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, second ed., 2002, https://doi.org/10.1137/1.9780898718027.

[10] N. J. Higham, S. Pranesh, and M. Zounon, *Squeezing a matrix into half precision, with an application to solving linear systems*, SIAM J. Sci. Comput., 41 (2019), pp. A2536–A2551, https://doi.org/10.1137/18M1229511.

[11] *IEEE Standard for Floating-Point Arithmetic, IEEE Std 754-2019 (revision of IEEE Std 754-2008)*, Institute of Electrical and Electronics Engineers, Piscataway, NJ, USA, July 2019, https://doi.org/10.1109/IEEESTD.2019.8766229.

[12] *BFLOAT16—Hardware Numerics Definition*, Intel Corporation, Nov. 2018, https://software.intel.com/en-us/download/bfloat16-hardware-numerics-definition. White paper. Document number 338302-001US.

[13] C.-P. Jeannerod and S. M. Rump, *Improved error bounds for inner products in floating-point arithmetic*, SIAM J. Matrix Anal. Appl., 34 (2013), pp. 338–344, https://doi.org/10.1137/120894488.

[14] P. Micikevicius, S. Oberman, P. Dubey, M. Cornea, A. Rodriguez, I. Bratt, R. Grisenthwaite, N. Jouppi, C. Chou, A. Huffman, M. Schulte, R. Wittig, D. Jani, and S. Deng, *OCP 8-bit floating point specification (OFP8)*, June 2023, https://www.opencompute.org/documents/ocp-8-bit-floating-point-specification-ofp8-revision-1-0-2023-12-01-pdf-1. Version 1.0.

[15] H. Ootomo and R. Yokota, *Recovering single precision accuracy from tensor cores while surpassing the FP32 theoretical peak performance*, The International Journal of High Performance Computing Applications, 36 (2022), pp. 475–491, https://doi.org/10.1177/10943420221090256.

[16] B. D. Rouhani, N. Garegrat, T. Savell, A. More, K.-N. Han, R. Zhao, M. Hall, J. Klar, E. Chung, Y. Yu, M. Schulte, R. Wittig, I. Bratt, N. Stephens, J. Milanovic, J. Brothers, P. Dubey, M. Cornea, A. Heinecke, A. Rodriguez, M. Langhammer, S. Deng, M. Naumov, P. Micikevicius, M. Siu, and C. Verrilli, *OCP microscaling formats (MX) specification*, Sept. 2023, https://www.opencompute.org/documents/

ocp-microscaling-formats-mx-v1-0-spec-final-pdf. Version 1.0.

[17] P. H. Sterbenz, *Floating-point computation*, Prentice-Hall Inc., Englewood Cliffs, N.J., 1974. Prentice-Hall Series in Automatic Computation.