Accurate and Reproducible Matrix Multiplication

Katsuhisa Ozaki

Department of Mathematical Sciences,
Shibaura Institute of Technology

SWIM2018, University of Rostock, July 25th, 2018.
Thanks to fruitful comments by S.M. Rump

Introduction

Our topic is matrix multiplication using numerical computations (floating-point arithmetic).

- Accuracy
- Performance (Speed)
- Reproducible

Introduction

Numerical results of matrix multiplication depend on

- the number of cores
- use of Fused Multiply-Add (FMA)
- block size

If the order of evaluation is changed, numerical results are also changed due to rounding errors.

Introduction

Let p be $2n$-vector.

For the case of a single core:

$$(((p_1 + p_2) + p_3) + \dots) + p_{2n}$$

For the case of two cores:

$$(p_1 + \cdots + p_n) + (p_{n+1} + \cdots + p_{2n})$$

Introduction

Let four floating-point numbers be $x1, y1, x2, y2$.

$$x_1 y_1 + x_2 y_2$$

If we have fused multiply-add:

$$\mathtt{fma}(x_2, y_2, \mathrm{fl}(x_1 y_1))$$

$\mathtt{fma}(a, b, c)$ produces an approximation of $ab + c$ with a single rounding.

Reproducibility

an ability to obtain a bit-wise identical FP result from multiple runs of the code on the same input data.

- ReproBLAS by Ahrens, Nguyen and Demmel
- ExBLAS by Iakymchuk, Collange, Defour and Graillat

We introduce reproducible algorithms for matrix multiplicatior

Importance of Reproducibility

These are problems for computer-assisted proof:

- update of software
- replace by a new computer

We may not check the proof again in the future. (although a mathematical proof can be checked by books anytime in the future)

Reproducible Matrix Multiplication

- Error-Free Transformation of Matrix Multiplication
  - Obtain Accurate Result

Our approach does not require new libraries.

We use (already existed) BLAS or PBLAS

Assume that neither overflow nor underflow occurs and divide and conquer method is not applied.

Notation

- $\mathbb{F}$: set of floating-point numbers (IEEE 754)
- $\mathbf{u}$: roundoff unit (ex. $\mathbf{u} = 2^{-53}$)
- $\mathrm{fl}(\cdots)$: computed result
- $A \in \mathbb{F}^{m \times n}$ and $B \in \mathbb{F}^{n \times p}$

The original paper

K. Ozaki, T. Ogita, S. Oishi, S.M. Rump: **Error-Free Transformation of Matrix Multiplication** by Using Fast Routines of Matrix Multiplication and its Applications, Numerical Algorithms, Vol. 59:1 (2012), pp. 95-118.

Improvement of Division

K. Ozaki, T. Ogita, S. Oishi, S. M. Rump: Generalization of Error-Free Transformation for Matrix Multiplication and its Application, Nonlinear Theory and its Applications, IEICE, Vol. 4:1 (2013), pp. 2-11.

Improvement using overflow

K. Ozaki, T. Ogita, S. Oishi: Error-free transformation of matrix multiplication with a posteriori validation, Numerical Linear Algebra with Applications, 23(5), 2016, pp. 931-946.

Example of Error-Free Transformation

We obtain $A^{(1)}, \underline{A}^{(2)} \in \mathbb{F}^{m\times n}$ and $B^{(1)}, \underline{B}^{(2)} \in \mathbb{F}^{n\times p}$ such that

$$A = A^{(1)} + \underline{A}^{(2)}, \quad B = B^{(1)} + \underline{B}^{(2)}, \quad A^{(1)}B^{(1)} = \mathrm{fl}(A^{(1)}B^{(1)})$$

and compute

$$AB = (A^{(1)} + \underline{A}^{(2)})(B^{(1)} + \underline{B}^{(2)}) = A^{(1)}B^{(1)} + A^{(1)}\underline{B}^{(2)} + \underline{A}^{(2)}B.$$

Splitting

Define $\beta \in \mathbb{F}$ and vectors $\sigma \in \mathbb{F}^m$ and $\tau \in \mathbb{F}^p$

$$\beta = \lceil (\log_2 \alpha - \log_2 \mathbf{u})/2 \rceil, \quad 1 \le \alpha \le n$$

$$\sigma_i = 2^{\beta} \cdot 2^{v_i}, \quad \tau_j = 2^{\beta} \cdot 2^{w_j}, \tag{1}$$

Here, the vectors $v \in \mathbb{F}^m$ and $w \in \mathbb{F}^p$ are

$$v_i = \lceil \log_2 \max_{1 \le j \le n} |a_{ij}| \rceil, \quad w_j = \lceil \log_2 \max_{1 \le i \le n} |b_{ij}| \rceil. \tag{2}$$

Error-Free Splitting

$A^{(1)}$ and $\underline{A}^{(2)}$ are obtained by

$$a_{ij}^{(1)} = \mathrm{fl}\left((a_{ij} + \sigma_i) - \sigma_i\right), \quad \underline{a}_{ij}^{(2)} = \mathrm{fl}\left(a_{ij} - a_{ij}^{(1)}\right) \tag{3}$$

Similarly, $B^{(1)}$ and $\underline{B}^{(2)}$ can be obtained by

$$b_{ij}^{(1)} = \mathrm{fl}\left((b_{ij} + \tau_j) - \tau_j\right), \quad \underline{b}_{ij}^{(2)} = \mathrm{fl}\left(b_{ij} - b_{ij}^{(1)}\right) \tag{4}$$

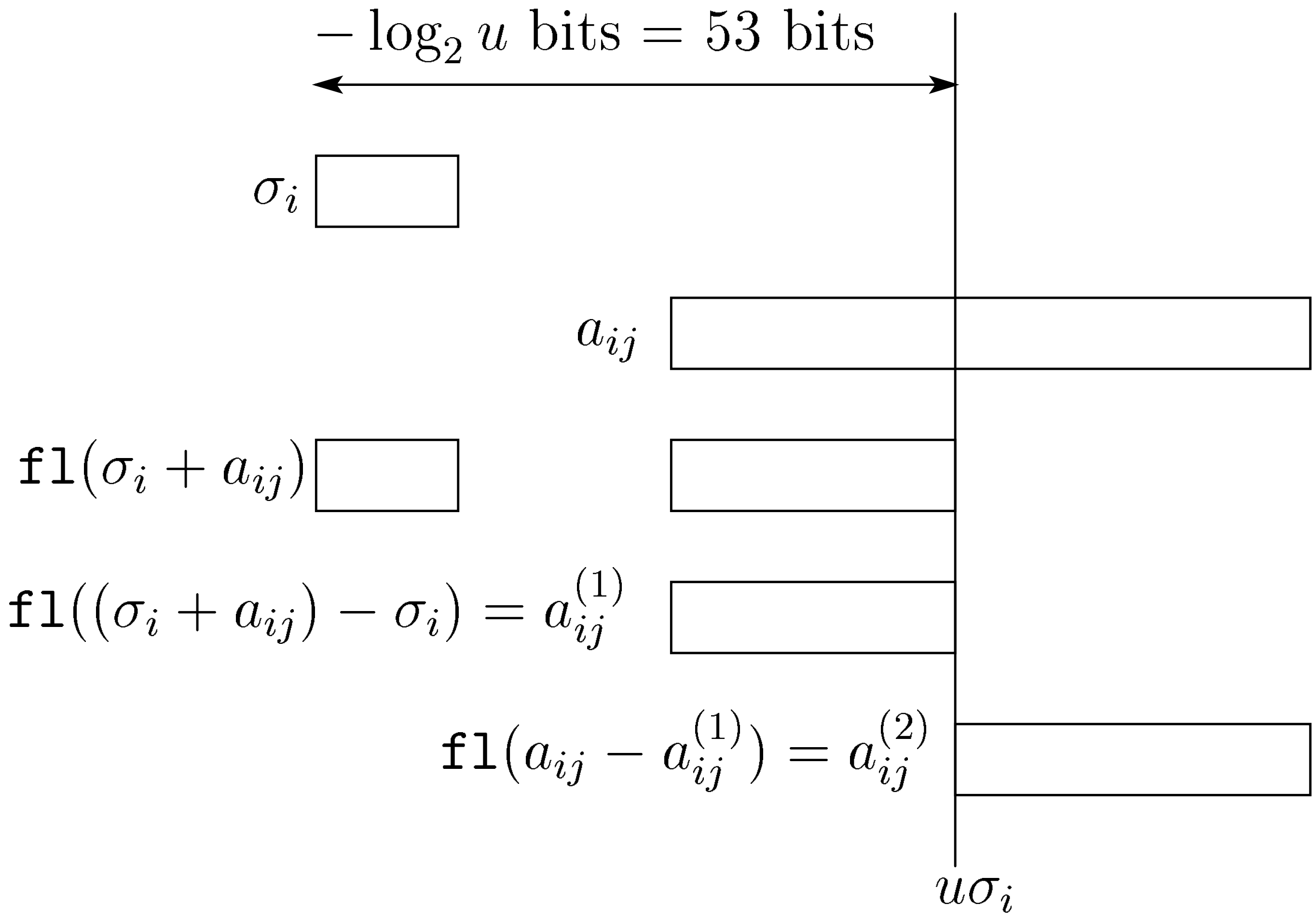

Figure 1: Image of the error-free splitting

Error-Free Splitting

$$
\begin{aligned}
A &= A^{(1)} + \underline{A}^{(2)}, && A^{(1)}, \underline{A}^{(2)} \in \mathbb{F}^{m\times n}, \\
\underline{A}^{(2)} &= A^{(2)} + \underline{A}^{(3)}, && A^{(2)}, \underline{A}^{(3)} \in \mathbb{F}^{m\times n}, \\
&\vdots \\
\underline{A}^{(k)} &= A^{(k)} + \underline{A}^{(k+1)}, && A^{(k)}, \underline{A}^{(k+1)} \in \mathbb{F}^{m\times n},
\end{aligned}
$$

Finally, $\underline{A}^{(k+1)}$ becomes the zero matrix.

Error-Free Transformation

Similarly, $B^{(1)}, \ldots, B^{(k)}$ can be obtained.

$$A = \sum_{i=1}^{k} A^{(i)}, \quad B = \sum_{j=1}^{l} B^{(j)}, \quad \mathrm{fl}(A^{(i)}B^{(j)}) = A^{(i)}B^{(j)}$$

and we have

$$AB = \mathrm{fl}(A^{(1)}B^{(1)}) + \cdots + \mathrm{fl}(A^{(k)}B^{(l)}).$$

Reproducible result can be obtained.

Error-Free Transformation

$$AB = \mathrm{fl}(A^{(1)}B^{(1)}) + \cdots + \mathrm{fl}(A^{(k)}B^{(l)}).$$

If we apply the accurate summation algorithm,

we obtain the best computed result.

Best means the correct rounding.

S.M. Rump, T. Ogita, and S. Oishi. Accurate floating-point summation part II: Sign, K-fold faithful and rounding to nearest. Siam J. Sci. Comput., 31(2):1269–1302, 2008.

Fast Reproducible Algorithms

For A and B, if

$$A = \sum_{i=1}^{5} A^{(i)}, \quad B = \sum_{i=1}^{5} B^{(i)},$$

then 25 matrix products are required.

We introduce simplified algorithms.

(This is not correct rounding)

Fast Reproducible Algorithms

We use

$$\begin{aligned} A &= A^{(1)} + \underline{A}^{(2)}, \quad B = B^{(1)} + \underline{B}^{(2)}, \\ AB &= A^{(1)}B^{(1)} + A^{(1)}\underline{B}^{(2)} + \underline{A}^{(2)}B \\ &= \mathrm{fl}(A^{(1)}B^{(1)}) + A^{(1)}\underline{B}^{(2)} + \underline{A}^{(2)}B \end{aligned}$$

We only use $\mathrm{fl}(A^{(1)}B^{(1)})$ such that

$$AB \approx \mathrm{fl}(A^{(1)}B^{(1)}).$$

Fast Reproducible Algorithms

We use

$$\begin{aligned} A &= A^{(1)} + A^{(2)} + \underline{A}^{(3)}, \quad B = B^{(1)} + B^{(2)} + \underline{B}^{(3)}, \\ AB &= \mathrm{fl}(A^{(1)}B^{(1)}) + \mathrm{fl}(A^{(1)}B^{(2)}) + \mathrm{fl}(A^{(2)}B^{(1)}) \\ &\quad + A^{(1)}\underline{B}^{(3)} + A^{(2)}\underline{B}^{(2)} + \underline{A}^{(3)}B. \end{aligned}$$

We only use three matrix products such that

$$AB \approx \mathrm{fl}(A^{(1)}B^{(1)}) + \mathrm{fl}(A^{(1)}B^{(2)}) + \mathrm{fl}(A^{(2)}B^{(1)}).$$

Fast Reproducible Algorithms

We use

$$A = A^{(1)} + A^{(2)} + A^{(3)} + \underline{A}^{(4)}, \quad B = B^{(1)} + B^{(2)} + B^{(3)} + \underline{B}^{(4)},$$

We only use six matrix products such that

$$\begin{aligned} AB &\approx \mathrm{fl}(A^{(1)}B^{(1)}) + \mathrm{fl}(A^{(1)}B^{(2)}) + \mathrm{fl}(A^{(2)}B^{(1)}) \\ &\quad + \mathrm{fl}(A^{(1)}B^{(3)}) + \mathrm{fl}(A^{(2)}B^{(2)}) + \mathrm{fl}(A^{(3)}B^{(1)}). \end{aligned}$$

Numerical Examples

$$A = \mathtt{randn}(n), \quad B = \mathtt{randn}(n)$$

Table 1: Comparison of Relative Error（$n = 10000$）

Methods	Average	Maximum
$A * B$	1.0127e-14	2.4751e-07
1 MM	1.0130e-05	2.0688e+02
3 MMs	2.4390e-12	3.0284e-05
6 MMs	5.3645e-17	5.7411e-12

Single Precision (Binary32)

- A, B: all elements are in binary32 $\Rightarrow$ binary64.
- $A = A^{(1)} + \underline{A}^{(2)}, \quad B = B^{(1)} + \underline{B}^{(2)}$
- compute $A^{(1)}B^{(1)}$
- the result is rounded to binary32

Single Precision (Binary32)

Table 2: Comparison of Relative Error ($n = 1000$)

Methods	Average	Maximum
$A * B$	1.8757e-06	5.8263e-02
1 MM	1.6181e-06	2.0905e-02

For matrix multiplication, computing time using binary64 is two times slower than that using binary32.

(There are exceptions for some GPUs)

Interval Matrix Multiplication

Reproducible algorithms for interval matrix multiplication.

$$\langle A_m, A_r \rangle, \quad \langle B_m, B_r \rangle$$

are mid-rad interval matrices.

Then,

$$\langle A_m, A_r \rangle \langle B_m, B_r \rangle \subseteq \langle A_m B_m,\ |A_m| B_r + A_r(|B_m| + B_r) \rangle$$

is well-known.

Interval Matrix Multiplication

$$\langle A_m, A_r\rangle\langle B_m, B_r\rangle \subseteq \langle A_m B_m,\ |A_m|B_r + A_r(|B_m| + B_r)\rangle$$

We can compute upper bounds without matrix multiplication.

For $|A| = A$ and $|B| = B$,

$$AB \le \min(Afe^T, eg^T B), \quad f_i = \max_j b_{ij}, \quad g_j = \max_i a_{ij}$$

where $e = (1, 1, \ldots, 1)^T$.

Interval Matrix Multiplication

$$\langle A_m, A_r\rangle\langle B_m, B_r\rangle \subseteq \langle A_m B_m,\ |A_m|B_r + A_r(|B_m| + B_r)\rangle$$

$$A_m = A_m^{(1)} + \underline{A}_m^{(2)}, \quad B_m = B_m^{(1)} + \underline{B}_m^{(2)}$$

We propose the following enclosure:

$$\langle A_m^{(1)} B_m^{(1)} + A_m^{(1)} B_m^{(2)} + \underline{A}_m^{(2)} B,\ |A_m|B_r + A_r(|B_m| + B_r)\rangle$$

$$\subseteq$$

$$\langle \mathrm{fl}(A_m^{(1)} B_m^{(1)}),\ |A_m^{(1)}||\underline{B}_m^{(2)}| + |\underline{A}_m^{(2)}||B| + |A_m|B_r + A_r(|B_m| + B_r)\rangle$$

Interval Matrix Multiplication

$$\langle A_m, A_r \rangle \langle B_m, B_r \rangle \subseteq \langle A_m B_m, \ |A_m| B_r + A_r(|B_m| + B_r) \rangle$$

$$A_m = A_m^{(1)} + A_m^{(2)} + \underline{A}_m^{(3)}, \quad B_m = B_m^{(1)} + B_m^{(2)} + \underline{B}_m^{(3)}$$

Mid-point and radius can be given by

$$\begin{aligned} \texttt{mid} &= A_m^{(1)} B_m^{(1)} + A_m^{(1)} B_m^{(2)} + A_m^{(2)} B_m^{(1)} + A_m^{(1)} B_m^{(3)} + A_m^{(2)} \underline{B}_m^{(2)} + A_m^{(3)} \\ \texttt{rad} &= |A_m| B_r + A_r(|B_m| + B_r) \end{aligned}$$

Interval Matrix Multiplication

$$\langle A_m, A_r\rangle\langle B_m, B_r\rangle \subseteq \langle A_m B_m,\ |A_m|B_r + A_r(|B_m| + B_r)\rangle$$

$$A_m = A_m^{(1)} + A_m^{(2)} + \underline{A}_m^{(3)}, \quad B_m = B_m^{(1)} + B_m^{(2)} + \underline{B}_m^{(3)}$$

Mid-point and radius can be given by

$$\begin{aligned} \mathtt{mid} &= \mathrm{fl}(A_m^{(1)} B_m^{(1)}) + \mathrm{fl}(A_m^{(1)} B_m^{(2)}) + \mathrm{fl}(A_m^{(2)} B_m^{(1)}) \\ \mathtt{rad} &= |A_m^{(1)}||B_m^{(3)}| + |A_m^{(2)}||\underline{B}_m^{(2)}| + |A_m^{(3)}||B| \\ &\quad + |A_m|B_r + A_r(|B_m| + B_r) \end{aligned}$$

Numerical Examples

We generate matrices for mid-point

$$A_m = \mathtt{randn}(n), \quad B_m = \mathtt{randn}(n)$$

and set matrices for radius as

$$A_r = c|A_m|, \quad B_r = c|B_m|$$

where c is a positive constant.

Numerical Examples

Table 3: Average of Radius ($n = 5000$)

Methods \ c	1e-15	1e-10	1e-05
Ogita-Oishi	3.35e-11	3.05e-06	3.05e-01
1 MM	1.13e-01	1.13e-01	4.18e-01
3 MMs	1.12e-07	3.16e-06	3.05e-01
6 MMs	3.06e-11	3.05e-06	3.05e-01

T. Ogita, S. Oishi: Fast Inclusion of Interval Matrix Multiplication, Reliable Computing 11:3 (2005), 191–205.

Conclusion

- We proposed reproducible algorithms for matrix multiplication
- This technique can be applied to interval matrix multiplication

Thank you very much
for your kind attention