FMA を巧みに利用した疑似 4,6,8 倍精度行列積の設計

尾崎 克久 [1], 小泉 透 [2]
[1] 芝浦工業大学，[2] 名古屋工業大学
e-mail : ozaki@sic.shibaura-it.ac.jp

1　概要

IEEE 754 [1] が定める浮動小数点演算による計算結果の正確性が不十分な場合，より高精度な計算の使用が考えられる．指数部の範囲は通常の浮動小数点数で問題はなく，仮数部の範囲を疑似的に拡大したい場合には，例えば Pair arithmetic (PA, [3])，Double-word arithmetic (DW 演算, [5]), Triple-word arithmetic (TW 演算, [4])、Quad-word arithmetic (QW 演算, [5]) などの方法が知られている．また，最近では TW 演算や QW 演算を簡略化したアルゴリズムも提案されている [6].

本講演では行列積（内積）を研究対象とし，これらの手法よりも低コストである計算方法を実現するために，Fused Multiply-Add を巧みに使用したアルゴリズムを設計し，数値実験によりその有用性を示す.

2　準備

$\mathbb{F}$ を IEEE 754 が定めるある固定精度の 2 進浮動小数点数の集合とする．本稿では，浮動小数点演算の際にオーバーフローやアンダーフローが発生しないことを仮定する．ここで，ある固定精度は binary32 や binary64 などを意味する．$a, b, c \in \mathbb{F}$ に対して $\texttt{fma}(a, b, c)$ は Fused Multiply-add により $ab + c$ の最近点である浮動小数点数を返すものとする．以下は [2] により提案された $ab + c$ の近似を $x + y, x, y \in \mathbb{F}$ で返すアルゴリズムである.

1: **function** $[x,y] = \textsc{FastTwoFMA}(a,b,c)$
2: 　　$x = \texttt{fma}(a, b, c)$;
3: 　　$y = \texttt{fma}(a, b, c - x)$;
4: **end function**

このアルゴリズムについて $c - x \in \mathbb{F}$ のとき，$x + y$ が $ab + c$ の良い近似となる．具体的には

$$|x + y - (ab + c)| \le u^2 |ab + c|$$

が成立する．ここで，u は単位丸めである．以後，$c - x \in \mathbb{F}$ を精度維持のための前提条件と呼ぶ.

3　提案手法

3 つの数 $x = x_1 + x_2$, $y = y_1 + y_2$, $z = z_1 + z_2$ に対して $xy + z$ の近似を $w_1 + w_2$ で返すアルゴリズムを以下に紹介する．ここでは $|x_1| \gg |x_2|$, $|y_1| \gg |y_2|$, $|z_1| \gg |z_2|$ を仮定する.

1: **function** $[w_1,w_2] = \mathrm{FMA2}(x_1,x_2,y_1,y_2,z_1,z_2)$
2: 　　$[w_1, w_2] = \texttt{FastTwoFMA}(x_1, y_1, z_1)$;
3: 　　$w_2 = w_2 + \texttt{fma}(x_1, y_2, \texttt{fma}(x_2, y_1, z_2))$;
4: **end function**

$|z_1| \geq 2|x_1 y_1|$ のとき，$w_1 + w_2$ は $xy + z$ の良い近似となる．上記には 6 演算を要する．PA で $xy + z$ の近似計算を行えば 12 演算，QD ライブラリに含まれる DW 演算（Sloppy 版）ならば 18 演算必要であるため，上記アルゴリズムは条件付きではあるが低コストである．

次に $|c| \geq |ab|$ のとき，$ab + c$ を $w_1 + w_2 + w_3$ と誤差なく変換するアルゴリズムを以下に紹介する．

1: **function** $[w_1,w_2,w_3] = \mathrm{FMA3}(a,b,c)$
2: $\quad e_1 = a * b;$
3: $\quad e_2 = \mathtt{fma}(a, b, -e_1);$
4: $\quad [w_1, e_3] = \mathtt{FastTwoSum}(c, e_1);$
5: $\quad [w_2, w_3] = \mathtt{FastTwoSum}(e_3, e_2);$
6: **end function**

各要素が $x_i = x_i^{(1)} + x_i^{(2)}$，$y_i = y_i^{(1)} + y_i^{(2)}$ であるベクトルの内積

$$\sum_{i=1}^{n} \left(x_i^{(1)} + x_i^{(2)}\right) \left(y_i^{(1)} + y_i^{(2)}\right)$$

を計算する際にアルゴリズム FMA2 を使用すると，精度維持のための前提条件が満たされない．そこで，

$$\left(c + \sum_{i=1}^{n} \left(x_i^{(1)} + x_i^{(2)}\right) \left(y_i^{(1)} + y_i^{(2)}\right)\right) - c$$

と，内積を第一項から逐次計算するときにはじめに大きな数を加え，最後に同じ値を引くことで精度維持のための前提条件を常に満たすように設計する．具体的な c の設定方法や，アルゴリズム FMA3 を疑似 TW 演算，疑似 QW 演算の設計に導入する方法などは講演時に紹介する．

謝辞　本研究は部分的に科研費 (23K28100) の御支援をいただいた．

参考文献

[1] IEEE Standard for Floating-Point Arithmetic, IEEE Std 754-2008, 2008.

[2] 小泉透, 津邑公暁, 入江英嗣, 坂井修一: 倍精度指数関数の高速な完全精度実装, 研究報告ハイパフォーマンスコンピューティング (HPC), Vol. 2023-HPC-192, 21, 1–12.

[3] S.M. Rump, M. Lange: Faithfully Rounded Floating-point Computations, ACM Transactions on Mathematical Software, Vol. 46 (2020), Article 21.

[4] N. Fabiano, J-M. Muller, J. Picot: Algorithms for Triple-Word Arithmetic, IEEE Transactions on Computers Vol. 68 (2019) , 1573–1583.

[5] D.H. Bailey, Y. Hida, X.S. Li, B. Thompson: QD: A Double-Double and Quad-Double Package, High Performance Computational Research, Lawrence Berkeley National Laboratory, 2000.

[6] 尾崎 克久, 今村 俊幸：Pair Arithmetic の拡張とその使い方について，研究報告ハイパフォーマンスコンピューティング (HPC), 2023-HPC-192(19), 1–8.

