多倍長精度行列乗算の高速化とその応用に関する研究*

Acceleration of Matrix Multiplication and Applications Using Multiple-Precision Floating-Point Arithmetic

打桐 大雅†
Taiga UTSUGIRI

1. はじめに

binary32，64 よりも長い桁数をサポートする手段としてマルチコンポーネント方式が存在し，DD(Double-Double, 106bit, 約 32 桁)型, TD(Triple-Double, 159bit, 約 49 桁)型，QD(Quadruple-Double，212bit，約 64 桁)が存在し，有名な実装として Bailey らの QD ライブラリ[1]が存在する．そのほかの方式として多数桁方式が存在し，任意の仮数部を設定でき高性能な浮動小数点ライブラリとして MPFR 型[2]が存在する．

一方で行列乗算の演算方法の 1 つに尾崎スキーム[3]が存在する．多倍長行列を桁落ちが発生しない binary32，64 に分割し，binary32，64 の高速な行列乗算を用いることで高速化を図る手法である．

本研究では，CPU では TD，DD，QD，MPFR 型では尾崎スキームの実装，GPU では TS 型の単純行列乗算，尾崎スキームの実装を実施する，さらに，TD，DD 型では連立一次方程式の LU 分解に適応し，その有用性を示す．

2. 最適化された多倍長精度線形計算ライブラリの現状

多倍長行列乗算における開発の現状を表 1 示す．表 1 より今回開発したものを紹介する．今回開発したものとして赤字で示すように CPU 上では TS，DD，TD，QD，MPFR 型における尾崎スキーム，GPU 上では TS，TD 型における単純行列乗算と尾崎スキームを用いた行列乗算の開発を行なった．

表 1 多倍長精度線形計算の最適化手法と研究一覧

xGEMM	CPU				GPU	
Opt.Method	None	AVX2	OpenMP	Ozaki Scheme	CUDA	Ozaki Scheme
DS	?	?	?	?	Mukunoki	Mukunoki
TS	?	?	?	Utsugiri	Utsugiri	Utsugiri
QS	?	?	?	?	?	?
IEEE754 Binary128	MPBLAS	?	MPBLAS	Mukunoki	Mukunoki	Mukunoki
DD	MPBLAS, BNCmatmul	Lis, MuPAT BNCmatmul	Lis,MuPAT, MPBLAS, BNCmatmul	Utsugiri	Mukunoki	Mukunoki
TD	BNCmatmul	BNCmatmul	BNCmatmul	Utsugiri	Utsugiri	Utsugiri
QD	MuPAT, MPBLAS, BNCmatmul	BNCmatmul	MuPAT, MPBLAS, BNCmatmul	Utsugiri	?	?
MPFR	MPBLAS, BNCmatmul	?	MPBLAS, BNCmatmul	Utsugiri	CUMP	?

MPLAPACK/MPBLAS https://github.com/mahonakata/mplapack
MuPAT Yagi, H et.al(2020)
Lis https://www.ssisc.org/lis/
CUMP https://github.com/skystar0227/CUMP
BNCmatmul https://na-inet.jp/na/bnc/
Mukunoki Mukunoki, D et.al.(2021)
Utugiri Utsugiri(2021-2023)

3. 尾崎スキームを用いた行列乗算の実装

尾崎スキームは，最適化された binary32，64 行列乗算(xGEMM)関数の高速性を生かすべく，無誤差変換技法の Split と酷似した浮動小数点の分割を行列単位で行う．

尾崎スキームの肝として多倍長精度行列 A，B を binary32，64 演算の xGEMM でも誤差が出ない程度の長さの binary32，64 行列A_1，A_2...とB_1，B_2...に分割する必要がある．こうして$AB = (\sum_i A_i)(\sum_j B_j)$ を展開して A_iB_j を xGEMM を用いて計算し，この結果を多倍長精度演算を用いて加算を行うことで精度の良い$C \approx AB$ を求めることができる．

TD 行列に対応した尾崎スキーム行列乗算のアルゴリズムを表 2 に示す．本研究では CPU では TD，DD，QD，MPFR 型，GPU では TS 型にて実装を行った．

表 2 TD 型行列乗算用の尾崎スキーム

Input: A, B　　:A, B は TD 型の正方行列
Output: C　　:C は TD 型の正方行列
1: $A^{(D)} = A$, $B^{(D)} = B$:$A^{(D)}$, $B^{(D)}$は double 型の n × n の正方行列
2: $\alpha = 1$
3: *While*($\alpha < 3$)
4: $M_A(i)_{1\le i\le n} = max_{1\le k\le n}|A^{(D)}{}_{i,k}|$ ：M_Aは n 次ベクトル
5: $M_B(j)_{1\le j\le n} = max_{1\le p\le n}|B^{(D)}{}_{p,j}|$ ：M_Bは n 次ベクトル
6: $T_A(i)_{1\le i\le n} = 2\text{^}(ceil(log2(M_A(i))) + ceil(\frac{53+log2(n)}{2}))$
：T_Aは n 次ベクトル
7: $T_B(j)_{1\le j\le n} = 2\text{^}(ceil(log2(M_B(j))) + ceil(\frac{53+log2(n)}{2}))$
：T_Bは n 次ベクトル
8: $S_A = T_A \cdot E^t$　：E = (1, 1, 1, …, 1)^t の n 次ベクトル
9: $S_B = E \cdot T_B{}^t$
10: $A_\alpha = (A^{(D)} + S_A) - S_A$
11: $B_\alpha = (B^{(D)} + S_B) - S_B$
12: $A = A - A_\alpha$, $B = B - B_\alpha$ TD 演算にて計算
13: $A^{(D)} = A$, $B^{(D)} = B$
14: $\alpha = \alpha + 1$
15: *End While*
16: $A_3 = A^{(D)}$
17: $B_3 = B^{(D)}$
18: *For*($\alpha = 1; \alpha < 4; \alpha++$)
19: *For*($\beta = 1; \beta < 4 - \alpha; \beta++$)
20: $C_\beta = A_\alpha \cdot B_\beta$ ：intel MKL の Dgemm にて計算
21: *End For*
22: $C \mathrel{+}= \sum_{k=1}^{4-\alpha} C_k$ ：TD 加算にて計算
23: *End For*

4. 多倍長浮動小数点演算のベンチマーク結果

本研究で使用したベンチマーク環境を表 3 に示す．今回使用した行列の要素を式 1，相対誤差の比較を式 2，コンパイルオプションなどを式 3 に示す．ruは

表 3 ベンチマークの実施環境

CPU	Intel Core i7 11700 (8C/16T)
Memory	32GB
GPU	NVIDIA GeForce RTX 3070
OS	Ubuntu 18.04.5 LTS
CUDA	11.0
Intel One API	2021.5.0

$a_{i,j},\ b_{i,j} = (ru - 0..5) \times exp\,(1.0 \times rn)$　式 1

$$\max_{1\le i,\ j\le n} \frac{\left|E^*_{i,\ j} - E_{i,\ j}\right|}{\left|E^*_{i,\ j}\right|}$$　式 2　真値　：$E^*_{i,\ j}$　実測値：$E_{i,\ j}$

$icpc - fp - model\ precse - O3 - lqd - qmkl\ (-lmpfr - lgmp)$　式 3

---

* 2022 年度修士論文概要
† 静岡理工科大学 大学院理工学研究科 システム工学専攻

[0, 1]の一様乱数であり，rnは標準正規分布に従う乱数とした．なお LU 分解では$a_{i,j}$，$b_{i,j} = ru$とした．

CPU における行列乗算のベンチマークの結果を図 1 に示す．左側が実行時間，右側が相対誤差を示している．比較としてそれぞれの精度にて Strassen+AVX2 を用いた行列乗算の測定を行った．まず右の相対誤差から説明する．縦軸が相対誤差，横軸が次元数を示している．図より TD 型では尾崎スキーム 7 分割と 8 分割の間，DD 型では 5 分割と同等，QD 型では 1536 次元までは 10 分割と同等，2048 以降では 10 分割と 11 分割の間に Strassen+AVX2 の相対誤差が入っていることが分かる．次に，左の実行時間について説明する．縦軸が実行時間，横軸が次元数を表している．図より，TD 型では最大の精度が出る 9 分割で約 6.9 倍,Strassen+AVX2 の精度が保証されている 8 分割で約 8.6 倍，DD 型では最大の精度が出る 6 分割で約 1.2 倍,Strassen+AVX2 の精度と同等の 5 分割で約 1.8 倍，QD 型では最大の精度が出る 12 分割で約 4.8 倍，Strassen+AVX2 以上の精度が出る 11 分割で約 5.7 倍 Strassen+AVX2 よりも高速であることが示された．

GPU における行列乗算のベンチマークを図 2(左)に示す．比較として DD 型，D+S(Double+Single)型の測定を行った．その結果，DD 型よりも約 9.3 倍，D+S 型よりも約 11.6 倍高速となった．TS 型では高速化のためシェアードメモリを使用した．その結果を図 2(右)に示す．シェアードメモリを使用したほうが約 2.3 倍高速となった．

CPU における TD 型行列乗算の応用として LU 分解に適応した．TD 型 LU 分解のベンチマークの結果を図 3 に示す．左側が実行時間，右側が相対誤差を示している．比較として Strassen+AVX2 を用いた LU 分解，単純 LU 分解の測定を行った．左側が実行時間，右側が相対誤差を示している．次元数が 512～1536 の 3 つの次元数でベンチマークを測定した．まず左側の実行時間より説明する．図より単純行列乗算よりも高速になるのは次元数が n = 1536 次元の時に，今回実施した尾崎スキームのすべての分割数よりも高速になった．次に，右の相対誤差を説明する．結果よりすべての次元数，MIN_DIM で単純行列乗算よりも精度が悪くなった．

5. まとめ

本研究では CPU における TD，DD，QD，MPFR 型において尾崎スキームの行列乗算を用いた結果，既存のものよりも高速化することができた．さらに LU 分解に適応することで TD 型においては有用性を確認できた．GPU では TS 型を作成し，既存のものよりも高速化することができた．

謝辞

本研究にあたっては，指導教員の幸谷智紀教授からは多くの助言とご指導を賜りましたことを厚く感謝申し上げます．なお本研究は科研費 20K11843 の助成を受けて行われているものです．

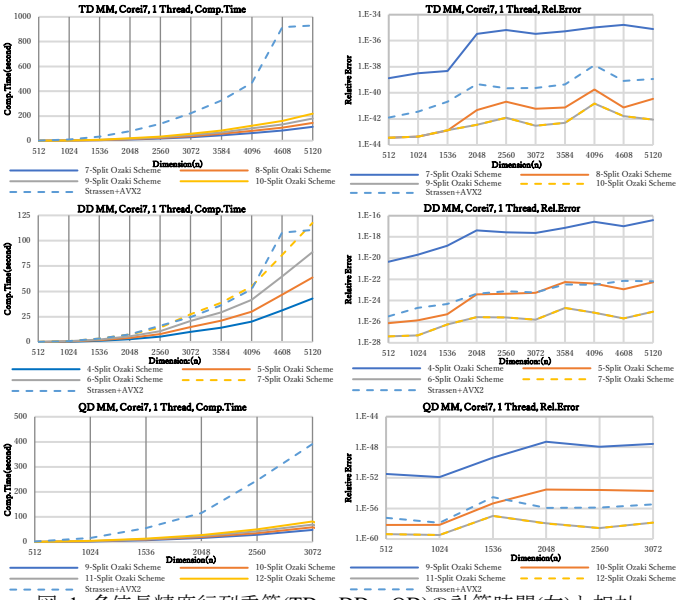

図 1 多倍長精度行列乗算(TD，DD，QD)の計算時間(左)と相対誤差(右)の比較

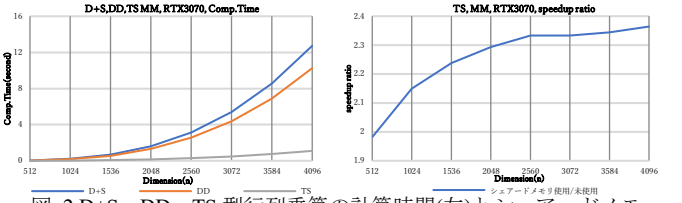

図 2 D+S，DD，TS 型行列乗算の計算時間(左)とシェアードメモリを用いた TS 精度行列乗算の性能向上比(右)

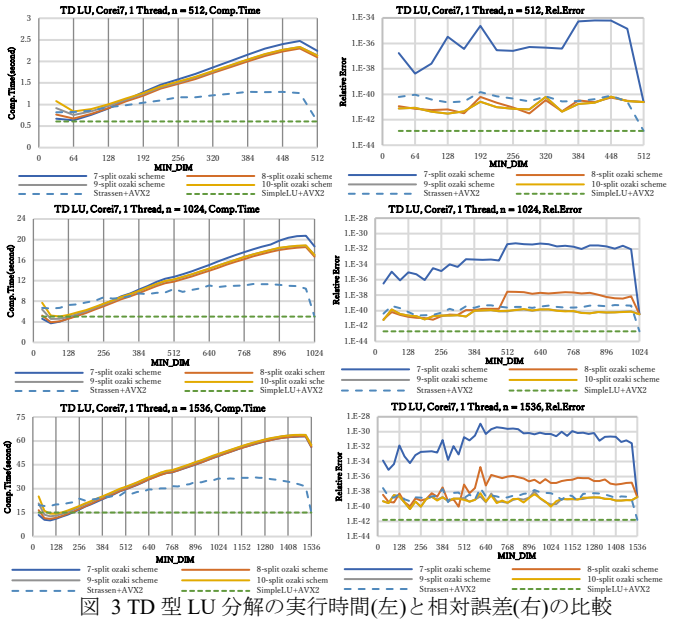

図 3 TD 型 LU 分解の実行時間(左)と相対誤差(右)の比較

参考文献

1） D.Bailey: QD, https://www.davidhbailey.com/dhbsoftware/.

2） The MPFR Library, https://www.mpfr.org/.

3） K.Ozaki et al.: Fast Algorithms for Floating-point Interval Matrix Multiplication, Journal of Computational and Applied Mathematics, Vol.236, pp.1795-1814(2012)