様　式　Ｃ－１９、Ｆ－１９－１、Ｚ－１９　（共通）

科学研究費助成事業　　研究成果報告書

令和　５ 年 １０ 月 ２４ 日現在

機関番号：３３８０３
研究種目：基盤研究(C)（一般）
研究期間：2020～2022
課題番号：２０Ｋ１１８４３

研究課題名（和文）マルチコアCPU向けに最適化された高速任意精度線形計算ライブラリの開発

研究課題名（英文）Development of efficient arbitrary precision numerical linear computation library optimized for multi-core CPUs

研究代表者

幸谷　智紀（Kouya, Tomonori）

静岡理工科大学・情報学部・教授

研究者番号：８０３１９１５２

交付決定額（研究期間全体）：（直接経費）　　2,200,000円

研究成果の概要（和文）：多倍長精度浮動小数点演算をサポートした高速な基本線形計算ライブラリを開発し，オープンソースとして公開した。独自のマルチコンポーネント方式の固定精度演算を含み，AVX2を用いた最適化も行って高速化している。多数桁方式の任意精度演算についてもCネイティブの関数呼び出しを行っており，C++実装より良いパフォーマンスを発揮している。行列乗算については尾崎スキームとStrassenアルゴリズムを実装しており，中間的な行列サイズと計算精度においては前者が後者より高速になることも確認し，論文としてまとめてある。

研究成果の学術的意義や社会的意義
我々の開発した多倍長精度最適化基本線形計算ライブラリは，標準的な多倍長精度線形計算ライブラリよりも高速であり，OpenMP，AVX2，尾崎スキーム，Strassenアルゴリズムといった，あらゆる最適化手法を取り入れている。よって，大規模な悪条件問題をより高速に解くことができ，スーパーコンピュータからコンシューマ向けのノートパソコンまで利用することができるものである。Pythonモジュールとしても実験的に実装しており，深層学習結果の検証や，可視化ツールとの連携も可能であり，敷居の高い多倍長精度計算環境をカジュアルに，高価な商用ソフトウェアを介することなく利用できる。

研究成果の概要（英文）：We have developed our original fast basic linear computation library supporting multiple-precision floating-point arithmetic, and already published it as open-source project. That includes multi-component-way fixed-precision arithmetic and optimized routines using AVX2, and multi-digit-way arbitrary precision arithmetic using C-native function call which can reduce overhead due to C++ class. For matrix multiplication, the Ozaki scheme and the Strassen algorithm have been implemented, so we have observed that the Ozaki scheme can be more efficient than Strassen algorithm in middle-precision arithmetic, and already published it as the international proceeding paper.

研究分野：高性能計算

キーワード：多倍長精度浮動小数点演算　基本線形計算　最適化　SIMD　OpenMP

1　研究開始当初の背景

現代社会はあらゆる領域に AI をはじめとする大量の浮動小数点演算を必要とするアプリケーションが求められるようになっている。マルチコア CPU や GPU がコンシューマレベルにまで手軽に利用できるようになっており，計算処理だけでなく，計算結果の保証も，これらのリソースを用いて実行することが求められる。binary64 を超える仮数部の精度を持つ多倍長精度浮動小数点演算も，計算結果の保証のために導入され，多倍長精度線形計算ライブラリ MPLAPACK/MPBLAS[3] が標準的な多倍長精度数値計算環境の基盤を提供するようになっている。

そのような状況下では，さらに高速な基本線形計算を提供する最適化ライブラリが求められるようになっている。標準的な最適化手法としては，CPU においては SIMD 命令の利用，OpenMP による並列化といったものがあるが，多倍長精度行列乗算に対しては，さらに分割統治法や尾崎スキームといったアルゴリズムによる最適化が可能である。x86 アーキテクチャで使用できる SIMD 命令としては AVX2 があり，これを用いて DD(Double-double, 106bits 仮数部長) 精度の疎行列対応も含む線形計算ライブラリとしては Lis[1] がある。

これらすべての最適化手法を使用でき，マルチコンポーネント方式による固定精度と，多数桁方式による任意精度計算の両方に対応した，MPBLAS を超える性能の新たな多倍長精度基本線形計算ライブラリを構築できる条件は整いつつあった。

2　研究の目的

binary64 より長い仮数部を持つ多倍長精度計算が必要な理由を概略的に示す。まず，ユーザが欲しているのは x という浮動小数点数で表現された入力値から $F(x)$ という値を浮動小数点演算を用いて求め，最終的には $F(x)$ が U 桁以上の有効桁数を保持していることである。数値計算の常識として，入力値には丸めによる初期誤差が混入していることを前提とする。使用する浮動小数点演算の仮数部の桁を $L(>U)$ とした時，$F(x)$ の有効桁数は U から R 桁だけ足りないという状況を考える。更に，L 桁計算でも精度を確保できるようアルゴリズムの変更が不可能，とする。

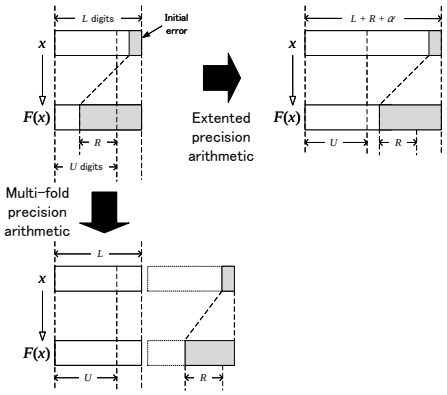

図 1　多倍長 (extended) 精度計算と多重 (multi-fold) 精度計算

$F(x)$ の計算アルゴリズムを変更しない限り，誤差の増加量は使用する浮動小数点演算の仮数部桁数に関わらず殆ど変化しない。従って，単純に仮数部の桁数を R より多少余裕を持たせて α 桁だけ増やした $L+R+\alpha$ 桁計算で実行でき，初期誤差がそれだけ小さくできるのであれば，U 桁以上の有効桁数を持つ $F(x)$ を得ることができる。これが多倍長精度計算を使用して精度を確保するアプローチである。

それに対し，Rump や荻田らの提唱した多重精度 (multi-fold) 計算 [6][7] は，無誤差変換技法を用いて演算で発生する丸め誤差を下位桁に追いやることで，初期誤差の増大を防ぐというアプローチを取る。無誤差変換の部分で計算量は増える点では多倍長精度計算と同じであるが，基本的に L 桁以上の浮動小数点演算は使用せず，多倍長精度演算とは異なり再正規化 (renormalization) プロセスを行う必要がないので，同様に無誤差変換技法を使用するマルチコンポーネント方式の多倍長精度計算より演算量を減らすことができるというメリットがある。

計算アルゴリズムを変更せずに初期誤差の増大を防ぐ，この 2 つのアプローチを概念図で示したのが図 1 である。

現在のところ，数値計算で用いられるすべてのアルゴリズム，特に非線形計算に対しては多重精度計算が適用できるわけではない。従って，例えば常微分方程式の初期値問題のように，多倍長精度計算を基本としつつ，多重精度計算が適用できる基本線形計算部分は多重精度計算を使用して計算時間を減らす，という使い分けを行う必要がある [2]。

我々のライブラリで取り入れるに至った，行列乗算アルゴリズムである尾崎スキーム [4] も，この多重精度計算のアプローチから生まれた技法であり，既存の binary32, binary64 精度の xGEMM の高速性を生かすことで，条件依存ではあるが，多倍長精度行列乗算の最適化に寄与する部分が大きい。

我々は要求される精度を持つ計算結果を保証し，かつ，最適な仮数部長で計算できることを目指した。従って，CPU 環境では binary64 を複数使用する DD, TD(Triple-double, 159bits), QD(Quadruple-double, 212bits) といったマルチコンポーネント方式の固定精度浮動小数点演算と，MPFR[5] 任意精度浮動小数点演算ライブラリを基盤とする最高性能を発揮する基本線形計算ライブラリを構築する。

3　研究の方法

本研究は，MBLAS より高性能な実行列乗算をはじめとする基本線形計算を，計算機環境に応じた最適化を行って安定的に最高速を実現することにある。(A) 多数桁方式の任意精度計算と，(B) マルチコンポーネント方式の比較的短い固定精度計算でそれぞれ分割して開発を行い，実行列乗算だけでなく，ベクトル計算，行列・ベクトル計算も含めて高速化を実現する。

以下，(A) と (B) の詳細を述べる。

(A) 多数桁方式の任意精度線形計算の最適化の追究　行列乗算高速化のネックになっている OpenMP 並列化の効果を最大限高める実装方式を追求する。MPFR の初期化関数のカスタマイズを行うことで，10% 程度の高速化が可能になることは確認できているが，より分割統治法の並列化の効果を高めるため，行列単位でメモリブロックをまとめる処理を行い，どの程度の効率化が可能になるかどうかを確認する。効果が上がることが確認できた時点で，ベクトル計算，行列・ベクトル計算にもこのメモリブロック方式を導入し，並列化の効果がどの程度上がるかを確認する。

(B) マルチコンポーネント方式の固定精度線形計算の最適化の追求
マルチコンポーネント方式の高速化については，既に Lis などに実装されている SIMD 化の効果が高いことが知られている。まず C コード化した `cdd_qd.h` に AVX256 の SIMD 命令を組み込み，基本線形計算が高速化できることを確認する。更に，分割統治法を用いた実行列乗算の高速化にどの程度寄与できるかを，ベンチマークテストによって確認する。また，無誤差変換技法を用いた 2 重精度算法の実装も行い，既存の Intel Math Kernel の BLAS 関数を用いた実装と比較を行って高速化の効果を確認する。

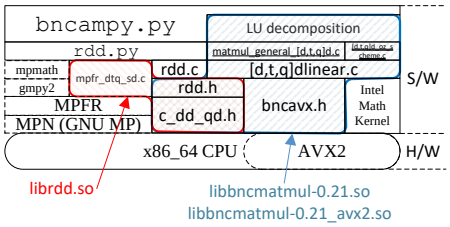

図 2　BNCmatmul ライブラリのソフトウェア階層

4　研究成果

今回の研究により，我々の，binary64 を超える仮数部の精度を持つ多倍長精度浮動小数点演算ライブラリ“BNCmatmul”は MPLAPACK/MPBLAS のパフォーマンスを超えるものを目指して構築したものであり，次のような特徴がある。

- 中核となるコーディングは ANSI C のスタイルで行っている。
- マルチコンポーネント方式の DD, TD, QD 精度をサポートしており，ベクトル・行列演算では AVX2 による高速化も行っている。
- MPFR を直接利用しており，mpreal クラスライブラリが持つオーバーヘッドが少ない。
- OpenMP による共有メモリ並列化をサポートしている。
- 行列乗算には単純 3 重ループ方式，ブロック化法，Strassen, Winograd アルゴリズムを実装している。

OpenMP 部分を除いた BNCmatmul の構成を図 2 に示す。

我々の DD, TD, QD 精度基本演算関数は c_dd_qd.h に C の inline 関数として実装されたものがベースになっており，これを rdd.h から MPFR 関数と同様の引数順に修正したマクロを介して呼び出して使用する。これを rdd.c に C の静的関数として定義したものを DL にしたものが librdd.so であり，Python のクラスはこの DLL 内の関数を呼び出して実装されている。

BNCmatmul も，rdd.h を用いて実装された行列乗算ライブラリであるが，前述したようにブロック化アルゴリズムと Strassen アルゴリズムを実装しており，これを使用するだけで単純行列乗算より高速になることは既に示している [9]。今回 AVX2 を用いて実装したのは，DD, TD, QD 精度の基本線形計算で，ブロック化，Strassen アルゴリズムの実装そのものは全く変更しておらず，AVX2 の高速化の恩恵は同じ関数を使用しながら受けることができる。我々の LU 分解はこれらの上で構築されており，後述のベンチマークテスト結果が示すように AVX2 による高速化も実現できている。

最適化した行列乗算の応用例と以上で定義した BNCmatmul(図 2) の関数は，DLL 化した libbncmatmul-0.21.so と AVX2 による高速化した libbncmatmul-0.21_avx2.so，どちらを呼び出しても同じ形式で使用できる。また，後述するように，これらを Python で利用するためのモジュールも試験的に作成し，有用性を確認してある。

以上，全てのソースコードは簡易的なマニュアルを付加して Github にアップロードしてある [8]。

また，現時点における多倍長精度行列乗算に関する最新の成果は，arXiv[*1]にアップロードしてある。

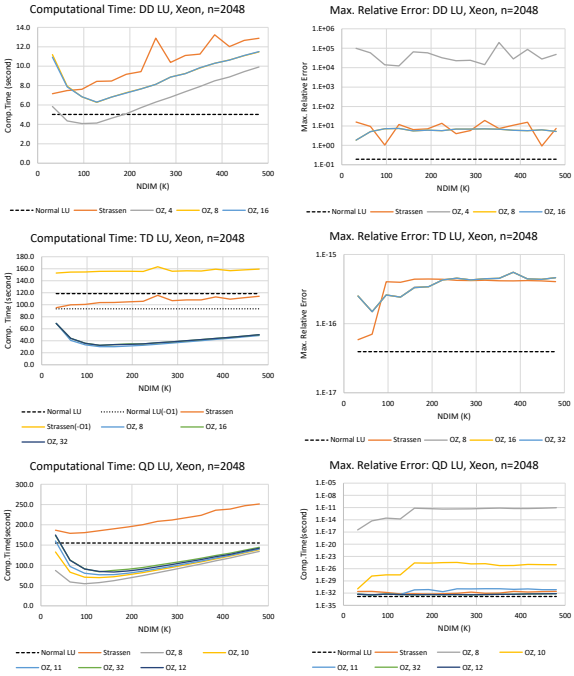

図 3　尾崎スキームを用いた LU 分解の計算時間 (左) と相対誤差 (右):DD 精度 (上), TD 精度 (中), QD 精度 (下)

■LU 分解への応用　Top500 をはじめ，種々のベンチマークテストに使用される直接法に用いられる LU 分解を我々の BNCmatmul を用いて実装し，その効用を Intel Xeon W-2295 上で計測した。現在の LAPACK 標準の LU 分解は高速な行列乗算が使用できることを前提に，一定幅 $K(=\mathrm{MIN\_DIM})$ をあらかじめ定めておき，K 列単位で矩形部分に行列乗算を実施してアップデートを行う。従ってこの K の値によって行列乗算のサイズが変化する。

列単位で計算を行う通常の LU 分解実装が一番高速で，Strassen 行列乗算を用いても，必ずしも性能向上が図れるわけではない。実際，DD, TD, QD 精度の LU 分解を実施し，計算時間 (左) と前進・後退代入を行って得た数値解の最大相対誤差 (右) に示したグラフを図 3 に示す。

DD 精度では最大一桁程度の精度しか得られない悪条件問題であり，Strassen アルゴリズムや尾崎スキームを用いるとさらに精度が 2〜3 桁落ちる。また計算時間も最大の精度を得るためには通常の LU 分解の計算時間を上回ることになる。TD, QD 精度計算になると，精度の差はさほどなくなり，計算時間も十分通常の LU 分解より小さくできる。ことに尾崎スキームを用いると，この問題の場合はどの K でも十分高速化を達成できていることが分かる。

では任意精度計算ではどうなるか？ MPFR256bits 計算と 512bits 計算で実行した結果を図 4 に示す。

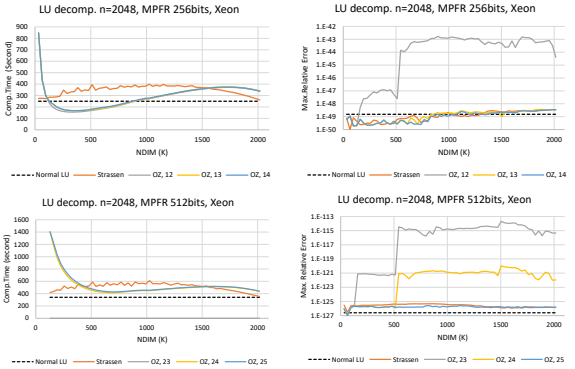

図 4　尾崎スキームを用いた LU 分解の計算時間 (左) と相対誤差 (右):MPFR 256bits(上), 512bits(下)

256bits 計算では小さい K の場合に通常の LU 分解より高速になる部分が見られるが，512bits になると全体的に低速になる。Strassen 行列乗算ではなしえない性能向上が尾崎スキームではみられるが，この問題に関する限り，

[*1] https://arxiv.org/abs/2301.09960

512bits 以上で有効活用できるところは少ないと思われる。

参考文献

[1] T. Kotakemori, S. Fujii, H. Hasegawa, and A. Nishida. Lis: Library of iterative solvers for linear systems. https://www.ssisc.org/lis/.

[2] Tomonori Kouya. Performance evaluation of an efficient double-double blas1 function with error-free transformation and its application to explicit extrapolation methods. In *2019 IEEE 26th Symposium on Computer Arithmetic (ARITH)*, pp. 120–123, 2019.

[3] MPLAPACK/MPBLAS. Multiple precision arithmetic LAPACK and BLAS. https://github.com/nakatamaho/mplapack.

[4] K. Ozaki, T. Ogita, S. M. Rump, and S. Oishi. Fast algorithms for floating-point interval matrix multiplication. *Journal of Computational and Applied Mathematics*, Vol. 236, pp. 1795–1814, 2012.

[5] MPFR Project. The MPFR library. https://www.mpfr.org/.

[6] S. M. Rump, T. Ogita, and S. Oishi. Accurate floating-point summation part I: Faithful rounding. *SIAM Journal on Scientific Computing*, Vol. 31, No. 1, pp. 189–224, 2008.

[7] S. M. Rump, T. Ogita, and S. Oishi. Accurate floating-point summation part II: Sign, k-fold faithful and rounding to nearest. *SIAM Journal on Scientific Computing*, Vol. 31, No. 2, pp. 1269–1302, 2008.

[8] T.Kouya. BNCmatmul. https://github.com/tkouya/bncmatmul.

[9] 幸谷智紀. 3 倍精度行列乗算の性能評価. 第 173 回 HPC 研究会, 2020.

５．主な発表論文等

〔雑誌論文〕　計8件（うち査読付論文　8件／うち国際共著　1件／うちオープンアクセス　2件）

１．著者名 Kouya Tomonori	４．巻 13378
２．論文標題 Acceleration of Multiple Precision Solver for Ill-Conditioned Algebraic Equations with Lower Precision Eigensolver	５．発行年 2022年
３．雑誌名 Lecture Notes in Computer Science (Springer)	６．最初と最後の頁 358～372
掲載論文のDOI（デジタルオブジェクト識別子） 10.1007/978-3-031-10562-3_26	査読の有無 有
オープンアクセス オープンアクセスではない、又はオープンアクセスが困難	国際共著 －

１．著者名 Utsugiri Taiga、Kouya Tomonori	４．巻 13378
２．論文標題 Acceleration of Matrix Multiplication Based on Triple-Double (TD), and Triple-Single (TS) Precision Arithmetic	５．発行年 2022年
３．雑誌名 Lecture Notes in Computer Science (Springer)	６．最初と最後の頁 406～423
掲載論文のDOI（デジタルオブジェクト識別子） 10.1007/978-3-031-10562-3_29	査読の有無 有
オープンアクセス オープンアクセスではない、又はオープンアクセスが困難	国際共著 該当する

１．著者名 Kouya Tomonori	４．巻 2022
２．論文標題 Optimization of mixed-precision iterative refinement using parallelized direct methods	５．発行年 2022年
３．雑誌名 2022 International Conference on Engineering and Emerging Technologies (ICEET)	６．最初と最後の頁 1～6
掲載論文のDOI（デジタルオブジェクト識別子） 10.1109/iceet56468.2022.10007230	査読の有無 有
オープンアクセス オープンアクセスではない、又はオープンアクセスが困難	国際共著 －

１．著者名 Kouya Tomonori	４．巻 12953
２．論文標題 Acceleration of Multiple Precision Matrix Multiplication Based on Multi-component Floating-Point Arithmetic Using AVX2	５．発行年 2021年
３．雑誌名 Lecture Notes in Computer Science	６．最初と最後の頁 202～217
掲載論文のDOI（デジタルオブジェクト識別子） 10.1007/978-3-030-86976-2_14	査読の有無 有
オープンアクセス オープンアクセスではない、又はオープンアクセスが困難	国際共著 －

１．著者名 Kouya Tomonori	４．巻 2021
２．論文標題 Acceleration of LU decomposition supporting double-double, triple-double, and quadruple-double precision floating-point arithmetic with AVX2	５．発行年 2021年
３．雑誌名 2021 IEEE 28th Symposium on Computer Arithmetic (ARITH)	６．最初と最後の頁 54～61
掲載論文のDOI（デジタルオブジェクト識別子） 10.1109/ARITH51176.2021.00021	査読の有無 有
オープンアクセス オープンアクセスではない、又はオープンアクセスが困難	国際共著 －

１．著者名 幸谷智紀	４．巻 29
２．論文標題 AVX2を用いたPythonプログラミング環境における多倍長精度線形計算高速化の試み	５．発行年 2021年
３．雑誌名 静岡理工科大学紀要	６．最初と最後の頁 49-56
掲載論文のDOI（デジタルオブジェクト識別子） なし	査読の有無 有
オープンアクセス オープンアクセスとしている（また、その予定である）	国際共著 －

１．著者名 Kouya Tomonori	４．巻 12253
２．論文標題 Performance Evaluation of Strassen Matrix Multiplication Supporting Triple-Double Precision Floating-Point Arithmetic	５．発行年 2020年
３．雑誌名 ICCSA 2020	６．最初と最後の頁 163～176
掲載論文のDOI（デジタルオブジェクト識別子） 10.1007/978-3-030-58814-4_12	査読の有無 有
オープンアクセス オープンアクセスではない、又はオープンアクセスが困難	国際共著 －

１．著者名 幸谷智紀	４．巻 28
２．論文標題 Python プログラミング環境における多倍長精度数値計算について	５．発行年 2020年
３．雑誌名 静岡理工科大学紀要	６．最初と最後の頁 23～31
掲載論文のDOI（デジタルオブジェクト識別子） なし	査読の有無 有
オープンアクセス オープンアクセスとしている（また、その予定である）	国際共著 －

〔学会発表〕　計6件（うち招待講演　0件／うち国際学会　0件）

1．発表者名
幸谷智紀

2．発表標題
AVX2を用いたマルチコンポーネント型多倍長精度直接法の性能評価

3．学会等名
第180回HPC研究会(SWoPP 2021)

4．発表年
2021年

1．発表者名
打桐大雅，幸谷智紀

2．発表標題
GPU における 3 倍精度浮動小数点数演算（Triple-Single）の性能評価

3．学会等名
電気・電子・情報関係学会 東海支部連合研究発表会

4．発表年
2021年

1．発表者名
打桐大雅，幸谷智紀

2．発表標題
コンシューマ向けGPUを用いた3倍精度(Triple-Single)行列積の性能評価

3．学会等名
第182回HPC研究発表会

4．発表年
2021年

1．発表者名
幸谷智紀

2．発表標題
AVX2を用いたマルチコンポーネント型多倍長精度行列乗算の高速化

3．学会等名
情報処理学会第178回HPC研究会

4．発表年
2021年

１．発表者名 幸谷智紀
２．発表標題 3倍精度演算を用いた高速行列乗算
３．学会等名 日本応用数理学会2020年度年会
４．発表年 2020年

１．発表者名 幸谷智紀
２．発表標題 SIMD命令を用いた3倍精度行列乗算の性能評価
３．学会等名 情報処理学会第176回HPC研究会（SWoPP2020）
４．発表年 2020年

〔図書〕　計1件

１．著者名 幸谷 智紀	４．発行年 2021年
２．出版社 講談社	５．総ページ数 272
３．書名 Python数値計算プログラミング	

〔産業財産権〕

〔その他〕

―

６．研究組織

	氏名 （ローマ字氏名） （研究者番号）	所属研究機関・部局・職 （機関番号）	備考

７．科研費を使用して開催した国際研究集会

〔国際研究集会〕　計0件

８．本研究に関連して実施した国際共同研究の実施状況

共同研究相手国	相手方研究機関