# MPACK (MBLAS/MLAPACK) 高精度 BLAS/LAPACK ライブラリの作成

Development of the MPACK library:
a multiple precision arithmetic version of BLAS and LAPACK

中田真秀[1)]
Maho NAKATA

[1)]博士 (工学)　理化学研究所 情報基盤センター (〒 351-0198 埼玉県和光市広沢 2-1 理化学研究所情報基盤センター 2F, maho@riken.jp)

We have been developing a multiple precision version of BLAS and LAPACK called MPACK (MBLAS/MLAPACK). The MPACK handles vector-vector, matrix-matrix operatons, and solves eigenvalue problems, etc. The MPACK has following features: (i) providing an application programming interface (API) which is a smooth extension to BLAS and LAPACK, (ii) portable; do not depend on multiple precision arithmetic libaries and operating systems, (iii) written in C++ instead of FORTRAN77, (iv) free software; licensed under GNU Lesser General Public License and is available at `http://mplapack.sourceforge.net/`. Currently 76 MBLAS routines and 90 routines out of 666 LAPACK routines are completed. We evaluated numerical performance of MPACK as well.

***Key Words :*** *multiple precision arithmetic, BLAS, LAPACK, MBLAS, MLAPACK*

## 1. はじめに

コンピュータを用いた数値計算による線形代数は数学、工学、科学、産業への幅広い分野への応用を持つ重要な基礎分野である。具体的には、行列積であったり、線形連立一次方程式を解いたり、行列の対角化、特異値分解などの操作をコンピュータ上で行う。これらの操作を、どれだけ規模の大きい問題を解くか、どれだけ速く解くか、どれだけ正確に解くか。この三つを人類は追求してきた。我々はこの三つ目、どれだけ正確に解くか、つまり線形代数演算の正確さ、品質に興味をもっている。特に行列の条件数が大きくなる場合や解きたい系が大きくなる場合など、演算をより正確に行わないと誤差が大きくなり、解が無意味になることがある。例えば半正定値計画法を 8 桁以上の有効桁で正確に解くためには、一般に高精度計算が必須である [1]。高精度線形代数演算の研究は過去に多くある。特にライブラリを作成しようとするものに、幸谷による BNCpack [2]、緒方らによる ALSQUAD [3] などがあるが、次のような問題がある。 BNCpack は C で実装されるのでプログラムが必要以上に複雑化する。 ALSQUAD は高速であるが、擬似四倍精度のみのサポートかつバイナリ配布のみであり、利用者は限定される。いずれも、幅広く使われている著名な線形代数ライブラリ BLAS [4] や LAPACK [5] とは全く違うスタイルでプログラムが必要で負担が大きい。これを踏まえ、我々は高精度線形代数演算ライブラリである MPACK (MBLAS/MLAPACK) を開発してきた [6]。ベクトル積、行列積、対角化などが行える。 BLAS [4] および LAPACK [5] を元にした多倍長精度版であり、特徴は (i) 参照実装の提供。 BLAS/LAPACK の自然な拡張となっている、 (ii) 高い移植性。多倍長精度演算ライブラリや OS になるべく依存せず、現在任意精度、擬似 8 倍、擬似 4 倍精度での計算が可能。 (iii) 高い可搬性。 C++ で実装し、従来の資産を機械的な置き換えに近い形で多倍長精度化が可能。 (iv) 自由なソフトウェア。 GNU Lesser General Public License 採用し、 `http://mplapack.sourceforge.net/` から入手可能である、などである。最新版は 0.6.4 で、 MBLAS 部は全 76 ルーチン、 MLAPACK 部は全 666 ルーチン中 90 ルーチンが完成した。これは半正定値計画法の多倍長精度版の作成から派生したプロジェクトである [1], [7]。本論文では MPACK の概要の説明と性能評価を行った。

## 2. 浮動小数点表現

コンピュータ上広く用いられる数の表現方法に 2 進数での浮動小数点表現がある。典型的には

$$(-1)^s \times 2^e \times m, \tag{1}$$

となっている。 $s$ は符号、つまり 0 または 1、 $e$ は指数、 $m$ は次の式で表される数字列である。

$$m = d_0.d_1d_2\cdots d_{p-1} \tag{2}$$

ここで $d_i$ は、0 または 1、$p$ は有効桁数である。表現や算術の定義の仕方は多様であるが、754-1985 IEEE Standard for Floating-Point Arithmetic [8] という規格をほとんどのシステムで採用することで (2008 年版が発効したが未だ採用システムは少ない)、再現性が高く品質の高い数値計算プログラムを安定して組めるようになっている。

## 3. BLAS と LAPACK

コンピュータ上での線形代数演算は様々な実装があるが、その中で最も広く使われているライブラリに、BLAS および LAPACK がある [4], [5]。BLAS の整備は 1970 年代後半から始まった。BLAS は Blas Linear Algebra Subprograms の略であり、ベクトルの内積、行列積など、基本的な線形代数演算を定義し Application Program Interface (API) 提供のための参照実装を目的とする。LAPACK は BLAS を内部で多用しつつ、連立一次方程式、最小二乗法、固有値問題、特異値分解などを行うことができる。歴史的には EISPACK や LINPACK を前身とする。BLAS, LAPACK はこれに加え実装それ自身が文書とできるほどの整然とした実装も特徴としてあげられる。また、これを元にした高速な実装はいくつも存在する。

## 4. 多倍長精度演算の必要性

近年のコンピュータのハードウェアの進歩で、数 G バイトを越える巨大な密行列の取扱いも容易になってきた。計算速度も高速になった。広く使われる IEEE754 の double precision (以下 IEEE double) の演算速度も、ピーク性能で Core i7-975 では 55.36GFlops (Floating point number Operations Per Second) であり、nVidia の Tesla C2050 も 515GFlops とどちらも驚異的である。IEEE double の有効桁は約 16 桁であるため次のような場合の計算結果が不正確になることがある。

1. 行列などが巨大になり演算回数が増える場合
2. 行列の条件数が高い連立一次方程式を解きたい場合
3. 数値安定性を要求する場合
4. 数学的な定理の確認をしたい場合

などなど。多倍長精度計算を行えば大部分解決すると考えられる。これは大雑把には式 (1) で定義された有効桁 $e$ の範囲や $p$ を大きくした計算である。著名なライブラリに GMP [9]、MPFR [10]、QD [11] や exflib [12] などがあり広く使われている。

## 5. MPACK で使う多倍長精度ライブラリ

現在 GMP [9] および QD [11] を用いている。GMP は式 (1) の $p$ を任意に指定できるが、64 の倍数になるように伸ばされる。$e$ は環境によって違うが 32 また 64 ビットの符号付き整数で表される。QD では、二つの精度計算が用意されている。IEEE double を 4 個 (QD 型) または 2 個 (DD 型) 並べて

$$a = (a_0, a_1, a_2, a_3)$$

一つの数とし、その上での演算等を定義する。浮動小数点演算のみで定義できるので実装が簡単、高速とされる。QD 型では約 64 桁、DD 型では約 32 桁の精度がある。

## 6. MPACK(MBLAS/MLAPACK) について

MPACK(MBLAS/MLAPACK) の特徴、品質保証、開発方法について述べる。

### (1) 参照実装の提供

BLAS や LAPACK の考え方に基づき、参照実装を提供することを主眼に置く。高い品質のルーチンを提供し、計算環境の向上を目指している為である。このため低速であるが、これを元にした高速な実装は可能である。既に椋木らによって GPGPU 上一部実装したものが存在する [13]。実装には C++ を選択したが、これは多倍長精度演算を C と同じように行いたいからだけであり、高速化を踏まえ C++ の特徴をなるべく使わず、C と互換性を保つような実装をしている。

### (2) 高いプラットフォーム非依存性

MPACK は OS や多倍長精度計算ライブラリになるべく依存しない形で書かれている。Windows, MacOSX, Linux, FreeBSD の四つのプラットフォームで動作確認をしている。多倍長演算ライブラリは、現在 0.6.4 では GMP(任意精度), QD(QD 型、DD 型) に対応しており、共通コードをそれらで複数回コンパイルする。尚、BLAS および LAPACK に必要な型は `REAL`, `COMPLEX`, `LOGICAL`, `INTEGER` の四つである。たとえば、多倍長精度演算ライブラリによって多倍長変数の型は `mpf_class`, `qd_real`, `dd_real` などとなるが、それらを `typedef` で使い分け、違いを吸収した。C との互換性および高速化の為テンプレートの利用は避けた。

### (3) 可搬性の高さ

`double` のかわりに多倍長実数クラスである `mpf_class`、`qd_real` などと置き換えればよい。`double` からのほぼ機械的な置き換えで対応可能である。半正定値計画法ソルバー SDPA について、その多倍長精度版である SDPA-GMP, -QD および -DD はそのように作成した [7]。

### (4) 自由の高さ: 無料、改変、再配布可能

`http://mplapack.sourceforge.net/` から入手できる。ライセンスは GNU Lesser General Public License を採用した。将来的には 2 条項 BSD ライセンスに改める予定である。

### (5) ルーチンの命名規則

BLAS および LAPACK のルーチン命名規則には接頭辞 “s”, “d”, “c”, “z” などがありそれぞれ単精度、倍精度、単精度複素数、単精度実数用を表す。 MPACK では接頭辞を実数を “R”、複素数を “C” と、多倍長精度計算クラスによらず統一した。 C++ では関数の多重定義により引数の違いで同名の関数をもてるからである。以下変換例を挙げる。

- daxpy, zaxpy → Raxpy, Caxpy
- dgemm, zgemm → Rgemm, Cgemm
- dsterf, dsyev → Rsterf, Rsyev
- dzabs1, dzasum → RCabs1, RCasum

### (6) 品質保証

計算の品質は多倍長精度で行ったからといって保証されるわけではない。 BLAS および LAPACK の信頼性を担保にし、次のように行った。 MBLAS および MLAPACK に様々な値を入力し、それぞれ BLAS および LAPACK との出力結果を比較、十分小さかった場合に正確だと判断した。明らかに不十分であるが、 MBLAS の演算は代数的処理のみであるから信頼はおけると判断した。また、エラー出力は BLAS の `xerbla` を横取りするような処理を行い、同じ値が出るというように確認した。 MLAPACK は LAPACK 同様、反復処理を行い収束させる操作も入っており、代入、比較するだけでは本質的には品質保証にはならない。特に有効桁である 16 桁を越える領域ではそうである。品質保証は各ルーチン個別の問題となると思わる。応用研究上で顕在化することも多く、系統的な品質保証は将来的な課題である。

### (7) 開発の進め方について

開発は次のように行った。 BLAS、 LAPACK のルーチンをとってきて `f2c` というツールを使い FORTRAN から C へ変換した。そのままでは可読性が全く無いので、 100 行程度の `sed` スクリプトを通し、おおまかに読める程度まで変換後、手作業で変換した。 `GO TO` やループ構造はほぼそのまま保持することにした。上記のように品質保証用プログラムを作成し、品質保証し、 BLAS、 LAPACK との一致がよければ完成とみなした。現在 MLAPACK を含め全てのルーチンは変換が終わっている。

## 7. 性能評価

`Raxpy`，`Rgemv`，`Rgemm`，`Rsyev` について性能評価した。計算は CPU: Intel Core i7 920 (2.67GHz, キャッシュ 8Mbytes、公称 42.56GFlops)、メモリ: 12Gbytes DDR3 1066MHz (10.667G/sec) という構成のマシン上で行った。ソフトウェアは、 Ubuntu 9.10 amd64, gcc4.4.1, GotoBLAS2-1.13 を BLAS および LAPACK として用いた。 GMP 型には 1024 ビットの仮数を与えた。これは約 308 桁の精度をもつ。繰り返すが DD 型では約 32 桁の精度、 QD 型では約 64 桁の精度をもつ。

### (1) Raxpy

`Raxpy` は、 BLAS の `daxpy` に相当する。 `incx = incy = 1` とし、 $n$ は 1 千万または 1 億と L3 キャッシュに収まらないようにした。一回の `Raxpy` 演算には $2n$Flops かかる。 DD 型は 250MFlops と GotoBLAS のほぼ 17% の性能である。メモリバンド幅が律速になるため、 DD 型、 QD 型演算はさらに加速できると思われる (表–1)。 GMP 型は QD 型よりも精度が高いにも関わらずほぼ同等のパフォーマンスを持っている。 OpenMP でマルチスレッド計算も行った。 2 倍から 3 倍高速化されたがコア数には比例しなかった。

**表–1 `Raxpy` パフォーマンス測定結果。速度の括弧内は OpenMP によるマルチスレッド計算**

| ライブラリ (有効桁) | 速度:Flops (OpenMP) |
|---|---|
| DD 型 (32) | 136(250)M |
| QD 型 (64) | 14.2(41.2)M |
| GMP 型 (308) | 14.2(38.8)M |
| GotoBLAS(16) | 1.5G |

### (2) Rgemv

`Rgemv` は BLAS の `dgemv` に相当する。 $n = m$ は 1 千又は 1 万として L3 キャッシュに収まらないようにし、 `incx = incy = 1` とした。一回の `Rgemv` 演算にはほぼ $2n^2$Flops かかる。今回も DD 型は `Raxpy` とほぼ同じパフォーマンスであるが、 GotoBLAS は約 3 倍高速であった。 DD 型、 QD 型、 GMP 型はほぼ `Raxpy` と同じパフォーマンスであったので、キャッシュはほぼミスヒットしている考えられる (表–2)。

**表–2 `Rgemv` パフォーマンス測定結果**

| ライブラリ (有効桁) | 速度:Flops |
|---|---|
| DD 型 (32) | 140M |
| QD 型 (64) | 13.9M |
| GMP 型 (308) | 14.0M |
| GotoBLAS(16) | 3.8G |

### (3) Rgemm

`Rgemm` は BLAS の `dgemm` に相当する。 $n = m = k$ は 1 千又は 1 万として L3 キャッシュに収まらないようにし、 `incx = incy = 1` とした。一回の `Rgemm` 演算に

はほぼ $2n^3$Flops かかる。この演算はキャッシュ性能に依存し、メモリのバンド幅に依存しないため、GotoBLAS は理論性能値に近いスピードで計算する。多倍長精度版は Raxpy や Rgemv とほぼ同じ性能で、同様にキャッシュはミスヒットしている考えられる (表 –3)。

**表–3 Rgemm パフォーマンス測定結果**

| ライブラリ (有効桁) | 速度:Flops |
|---|---|
| DD 型 (32) | 141M |
| QD 型 (64) | 13.9M |
| GMP 型 (308) | 14.4M |
| GotoBLAS(16) | 42.5G |

### (4) Rsyev

Rsyev は LAPACK の dsyev に相当し、対称行列の固有値、固有ベクトルを求める。表 –4 に 1000 次元の対称行列の全固有値及び固有ベクトルを求めるのにかかった時間をまとめた。DD 型は約 60 倍、GMP 型は約 2000 倍遅い。QD 型はスケーリングが多発したため、計算時間が GMP 型と比較して増えている。DD 型、QD 型では指数部の範囲が IEEE double より狭くなっており、アンダーフローが起こりやすくなっているのが原因だと思われる。

**表–4 Rsyev パフォーマンス; 1000 次元の行列の全固有値、全固有ベクトルを求める時間**

| ライブラリ (有効桁) | 速度: 秒 |
|---|---|
| DD 型 (32) | 112 |
| QD 型 (64) | 15088 |
| GMP 型 (308) | 3666 |
| GotoBLAS(16) | 1.83 |

## 8. 結論と今後の課題

多倍長精度版の BLAS および LAPACK である MPACK 最新版 0.6.4 の紹介と性能評価を行った。参照実装の提供を主とし、MBLAS の実装および MLAPACK の 90 のルーチンが完成している。C++ で書かれ、多倍長精度演算ライブラリやプラットフォームには依存せず、過去の資産活用も容易である。パフォーマンスは演算タイプによらずほぼ DD 型で 140MFlops、QD 型、GMP 型 (1024 ビット仮数) で 14MFlops であった。現在行っているのは、(i) MPFR 対応; よりよい品質保証と誤差評価、(iii) LAPACK ルーチンの充実、(iv) マルチコアや GPGPU などを用いた高速化、および (v) 並列計算向け ScaLAPACK の多倍長化などである。

**謝辞:** 研究の一部は理化学研究所基礎科学特別研究員制度および科学研究費補助金基盤研究 (B)(一般) 21300017「マルチプラットフォームの大規模数値シミュレーションを支援するフレームワークの構築」による。

### 参考文献

1) Nakata, M. et al. :Variational calculation of second-order reduced density matrices by strong $N$-representability conditions and an accurate semidefinite programming solver, *Journal of Chemical Physics*, Vol.128, 164113, 2008.
2) Kouya, T. :BNCpack, Basic Numerical Calculation PACKage, http://na-inet.jp/na/bnc/bnc.html, 2002-2010.
3) 緒方 隆盛, 久保 克維, 武井 利文: 高速 4 倍精度演算パッケージ ASLQUAD の概要, NEC 技報, Vol. 61, pp.54-57, 2008.
4) Lawson, C. L. et al. :Basic Linear Algebra Subprograms for FORTRAN usage, *ACM Trans. Math. Soft.*, Vol.5, pp.308-323, 1979.
5) Anderson, E. et al.: LAPACK Users' Guide Third Ed., Society for Industrial and Applied Mathematics, 1999.
6) Nakata, M: The MPACK (MBLAS/MLAPACK); a multiple precision arithmetic version BLAS and LAPACK, http://mplapack.sourceforge.net/, version 0.6.4, 2010.
7) Yamashita, M. et al. :A high-performance software package for semidefinite programs: SDPA 7, submitted.
8) IEEE: 754-1985 IEEE Standard for Floating-Point Arithmetic, 1985.
9) Granlund, T. et al. : GNU Multiple Precision Arithmetic Library, http://gmplib.org/, 2002.
10) Fousse, L. et al. : MPFR: A Multiple-Precision Binary Floating-Point Library with Correct Rounding, *ACM Trans. Math. Soft.*, Vol.33, pp.13:1-13:15, 2007.
11) Hida, Y. et al. : Quad-Double Arithmetic: Algorithms, Implementation, and Application, *Technical Report LBNL-46996, Lawrence Berkeley National Laboratory*, 2000.
12) Fujiwara, H. :exflib-extend precision floating-point arithmetic library, http://www-an.acs.i.kyoto-u.ac.jp/~fujiwara/exflib/, 2005-2009.
13) 椋木大地, 高橋大介 :GPU による 4 倍精度 BLAS の実装と評価, 情報処理学会研究報告, HPC, Vol.123, pp.13, 2009.