MPACK 0.6.0: 多倍長精度版のBLAS/LAPACKの作成

中田真秀

maho@riken.jp

理化学研究所情報基盤センター

線形計算研究会 2009/12/11

大枠

- 自己紹介
- MPACK(MBLAS/MLAPACK) の紹介
- ライブラリ作成の動機
- 現在の状況
- 今後の予定

自己紹介

- **中田真秀 (なかたまほ)**
- **理化学研究所 情報基盤センター**
- **基礎科学特別研究員** (2007-2010)
- **量子化学、理論化学**
- **博士 (工学)**
- http://accc.riken.jp/maho/

MPACK:多倍長精度版の BLAS/LAPACK

http://mplapack.sourceforge.net/

- MBLAS: BLAS(Basic Linear Algebra Subprograms) **の多倍長精度版。**
- MLAPACK: LAPACK (Linear Algebra PACKage) **の多倍長精度版。**
- **参照実装** (API) **の提供が第一目標。**
- LAPACK 3.1.1 **をベースとする。**
- **高い移植性**; **多倍長精度演算ライブラリや** OS **への非依存性。**
- C++**で書かれている。**
- LGPL; **誰でも自由に使えるライセンス。**

MPACK: 多倍長精度版の BLAS/LAPACK

http://mplapack.sourceforge.net/

- **最新版**: 0.6.0 (2009/11/24)。
- MBLAS **全** 76 **ルーチン完成。**
- MLAPACK **は** 50 **程度完成、**FORTRAN **から** C++**へ**: 668 **個完了。**
- **対角化、コレスキー分解、**LU **分解、逆行列を求めることなどが可能。**
- **多倍長精度計算ライブラリ**:GMP/QD/DD **が利用可能。**
- SDPA-GMP, -QD, -DD: powered by MPACK.
- 2007 **年中旬から開発開始した。**

MPACK (MBLAS/MLAPACK) に関する事実

http://mplapack.sourceforge.net/

- Google で"Multiple precision BLAS" で検索すると一番上に出てくる。
- page view 3069 hits, download 343

Date (UTC)	Rank	Total Pages [1]	Downloads	Project Web Hits	Tracker opened (closed)	Forum Posts
Dec 2009 [*]	N/D	79	11	258	0 (0)	0
Nov 2009	3,901	311	66	825	0 (0)	0
Oct 2009	2,013	325	55	746	0 (0)	0
Sep 2009	18,846	335	27	590	0 (0)	0
Aug 2009	82,634	138	28	438	0 (0)	0
Jul 2009	90,825	63	10	535	0 (0)	0
Jun 2009	68,481	125	19	473	0 (0)	0
May 2009	32,211	136	13	533	0 (0)	0
Apr 2009	10,449	174	24	388	0 (0)	0
Mar 2009	4,249	273	34	461	0 (0)	0
Feb 2009	3,177	647	39	407	0 (0)	0
Jan 2009	5,640	798	17	208	0 (0)	0

[*] Partial data: End of month not yet reached

[1] Calculated as sourceforge.net page views plus sflogo button impressions

きっかけ:量子化学

- 化学を第一原理的に研究したい。
- 電子と原子核からなる物質; 原子
- 原子と原子の結合; 分子
- 分子の組みかわり; 化学
- 量子論; シュレーディンガー方程式

きっかけ:密度行列の方法

- シュレーディンガー方程式は解くのが難しい。

$$H\Psi = E\Psi$$

- 変数の数をできるだけ落とした方法。
- 縮約密度行列変分法 [Journal of Chemical Physics, 114, 8282-8292 (2001)]

$$E = \min_{\gamma,\Gamma \in N\text{-rep.}} \left\{ \sum_{ij} v^i_j \gamma^i_j + \sum_{i_1 i_2 j_1 j_2} w^{i_1 i_2}_{j_1 j_2} \Gamma^{i_1 i_2}_{j_1 j_2} \right\}$$

- 半正定値計画; positive SemiDefinite Programming に帰着できる。

きっかけ:精度の必要性

- 量子化学の問題を半正定値計画法を使って解いていた。
- 半正定値計画法ソルバにはSDPA/SDPARAを使った。
- 大抵7~8桁の精度で事足りてた。
- いくつかの問題は精度が足りなかった。
  - 規模の大きな系
  - 電子相関の強い系

解決:多倍長精度計算

- 半正定値計画:精度を上げるのが困難。
- double のままで何とかしたい → 理論およびライブラリなど整備が必要 (面倒)

解決:多倍長精度計算

- 半正定値計画:精度を上げるのが困難。
- double のままで何とかしたい → 理論およびライブラリなど整備が必要 (面倒)

まずは多倍長精度計算で解決しよう

一つの結果:SDPA-GMP:GMP バージョンSDPA

- **半正定値計画法ソルバ** SDPA **をベース。**
- **多倍長精度演算ライブラリ**;GMP
- BLAS/LAPACK **など、行列演算の部分を** GMP(C++) **に書き換えた。**
- SDPA-GMP, SDPA-DD, QD **公開中**
  - Journal of Chemical Physics 128, 16, 164113 (2008).
  - http://sdpa.indsys.chuo-u.ac.jp/sdpa/download.html
  - SDPA-GMP **を使った他の論文**: Waki *et al*. (900 **桁必要**), Mittelmann *et al*., De Klerk *et al*.
  - **自由なソフトウェア**:GNU General Public License
- **小島政和先生「重要なソフトウェアです」**

飛躍:汎用性がある MPACK(MBLAS/MLAPACK) へ

- SDPA-GMP への初期の実装は汎用性がなかった。
- 他の分野、他の多倍長演算ライブラリへ応用できるようにしたい。
- 精度への要求 : 計算の高速化、多量化の次。

飛躍:汎用性がある MPACK(MBLAS/MLAPACK) へ

- SDPA-GMP への初期の実装は汎用性がなかった。
- 他の分野、他の多倍長演算ライブラリへ応用できるようにしたい。
- 精度への要求 : 計算の高速化、多量化の次。
  - 量子化学で精度が足りなくなってきている。

飛躍:汎用性がある MPACK(MBLAS/MLAPACK) へ

- SDPA-GMP への初期の実装は汎用性がなかった。
- 他の分野、他の多倍長演算ライブラリへ応用できるようにしたい。
- 精度への要求 : 計算の高速化、多量化の次。
  - 量子化学で精度が足りなくなってきている。
  - 実質的無限精度とし、定理の確認などが可能。

飛躍:汎用性がある MPACK(MBLAS/MLAPACK) へ

- SDPA-GMP への初期の実装は汎用性がなかった。
- 他の分野、他の多倍長演算ライブラリへ応用できるようにしたい。
- 精度への要求 : 計算の高速化、多量化の次。
  - 量子化学で精度が足りなくなってきている。
  - 実質的無限精度とし、定理の確認などが可能。
  - Google で調べると多倍長精度 BLAS/LAPACK への要求がいくつかあった。

過去のプログラムパッケージ

- BNCPack; T. Kouya; GMP, C **のみ**, **独自インターフェース、フリー**
- ASLQUAD; T. Ogata, K. Kubo and T. Takei, QD, **独自インターフェース、高速** (?)**、フリーじゃない**
- XBLAS; X. Li *et al.*, BLAS **の拡張、入力が倍精度、フリー**

何故にLAPACK/BLASか?

- 世界中で試されているBLAS/LAPACKの計算の品質は高い。
- 独自ルーチン作成は思わぬバグ混入のもととなる。
- バグのチェックもしやすい。
- これまでの資産活用も可能。
- 非常によく練られた、素晴らしいパッケージである
- 美しい、芸術品のようなソースコード

何故にLAPACK/BLASか?

- 世界中で試されているBLAS/LAPACKの計算の品質は高い。
- 独自ルーチン作成は思わぬバグ混入のもととなる。
- バグのチェックもしやすい。
- これまでの資産活用も可能。
- 非常によく練られた、素晴らしいパッケージである
- 美しい、芸術品のようなソースコード

BLAS/LAPACKにならうことにした

BLAS とは?

- The BLAS (Basic Linear Algebra Subprograms) are routines that provide standard building blocks for performing basic vector and matrix operations.
- The Level 1 BLAS perform scalar, vector and vector-vector operations, the Level 2 BLAS perform matrix-vector operations, and the Level 3 BLAS perform matrix-matrix operations. [1]
  - Level 1 : ${}^t\boldsymbol{x} \cdot \boldsymbol{x}$, ${}^t\boldsymbol{x} \cdot \boldsymbol{y}$, etc.
  - Level 2 : $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$, ${}^t\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$, etc.
  - Level 3 : $\alpha\boldsymbol{A}\boldsymbol{B} + \beta\boldsymbol{C}$ etc.

http://www.netlib.org/blas/faq.html [1]

LAPACKとは?

- LAPACK provides routines for solving systems of simultaneous linear equations, least-squares solutions of linear systems of equations, eigenvalue problems, and singular value problems. The associated matrix factorizations (LU, Cholesky, QR, SVD, Schur, generalized Schur) are also provided, as are related computations such as reordering of the Schur factorizations and estimating condition numbers. Dense and banded matrices are handled, but not general sparse matrices. In all areas, similar functionality is provided for real and complex matrices, in both single and double precision.

(http://www.netlib.org/lapack/faq.html より)

MPACK (MBLAS/MLAPACK) の開発方針

- **参照実装の提供、資産活用**; **ほぼ** BLAS/LAPACK **と同じ** API **の提供。**
- **品質保証**; **計算が正しいこと。**
- **汎用性の確保**; OS**、プロセッサ、多倍長精度演算ライブラリに依存しない。**
- **自由**; GNU Lesser General Public License**。**
- **開発の透明性**; SourceForge.net **で開発。**
- **言語**; FORTRAN **から** C++**へ移行。**

参照実装の提供; 資産活用; 命名規則

Prefix **の変化** float, double → “R”eal, complex, double complex → “C”complex.

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

参照実装の提供; 資産活用; 関数のコール例

違いは call by value or call by reference

MBLAS/MLAPACK

```
Rgemm("n", "n", n, n, n, alpha, A, n, B, n, beta, C, n);
Rgetrf(n, n, A, n, ipiv, &info);
Rgetri(n, A, n, ipiv, work, lwork, &info);
Rsyev("V", "U", n, A, n, w, work, &lwork, &info);
```

BLAS/LAPACK

```
dgemm_f77("N", "N", &n, &n, &n, &One, A, &n, A, &n, &Zero, C, &n
dgetri_f77(&n, A, &n, ipiv, work, &lwork, &info);
```

(C++/C **から** BLAS/LAPACK **を呼べる** lapack.h, blas.h **提供中**!)

MBLAS ソースコードから抜粋

Caxpy; axpy の複素数版

```
void
Caxpy(INTEGER n, COMPLEX ca, COMPLEX * cx, INTEGER incx, COMPLEX * cy, INTEGER i
{
    REAL Zero = 0.0;
    if (n <= 0)
        return;
    if (RCabs1(ca) == Zero)
        return;
    INTEGER ix = 0;
    INTEGER iy = 0;
    if (incx < 0)
        ix = (-n + 1) * incx;
    if (incy < 0)
        iy = (-n + 1) * incy;
    for (INTEGER i = 0; i < n; i++) {
        cy[iy] = cy[iy] + ca * cx[ix];
        ix = ix + incx;
```

MLAPACKソースコードから抜粋

Rsyev; 対称行列対角化ルーチン

```
        Rlascl(uplo, 0, 0, One, sigma, n, n, A, lda, info);
    }
//Call DSYTRD to reduce symmetric matrix to tridiagonal form.
    inde = 1;
    indtau = inde + n;
    indwrk = indtau + n;
    llwork = *lwork - indwrk + 1;
    Rsytrd(uplo, n, &A[0], lda, &w[0], &work[inde - 1], &work[indtau - 1],
        &work[indwrk - 1], llwork, &iinfo);
//For eigenvalues only, call DSTERF.  For eigenvectors, first call
//DORGTR to generate the orthogonal matrix, then call DSTEQR.
    if (!wantz) {
        Rsterf(n, &w[0], &work[inde - 1], info);
    } else {
        Rorgtr(uplo, n, A, lda, &work[indtau - 1], &work[indwrk - 1], llwork,
            &iinfo);
        Rsteqr(jobz, n, w, &work[inde - 1], A, lda, &work[indtau - 1], info);
    }
//If matrix was scaled, then rescale eigenvalues appropriately.
    if (iscale == 1) {
        if (*info == 0) {
```

MBLASの利用例

SDPA-GMP 7.1.2のsdpa_linear.cppより

```
    if (scalar==NULL) {
      scalar = &MONE;
      // scalar is local variable
    }
    // The Point is the first argument is "Transpose".
    Rgemm("Transpose","NoTranspose",retMat.nRow,retMat.nCol,aMat.nCol,
           *scalar,aMat.de_ele,aMat.nCol,bMat.de_ele,bMat.nRow,
           0.0,retMat.de_ele,retMat.nRow);
    break;
  case DenseMatrix::COMPLETION:
    rError("no support for COMPLETION");
    break;
  }

  return _SUCCESS;
```

MLAPACK利用例

SDPA-GMP 7.1.2のsdpa_linear.cppより

```
case DenseMatrix::DENSE:
  LWORK = 3*N-1;
  // "N" means that we need not eigen vectors
  // "L" means that we refer only lower triangular.
  Rsyev("NonVectors","Lower",N,aMat.de_ele,N, eigenVec.ele,workVec.ele,&LWORK,
  if (info!=0) {
    if (info < 0) {
      rMessage("getMinEigenValue:: info is mistaken " << info);
    } else {
      rMessage("getMinEigenValue:: cannot decomposition");
    }
    exit(0);
    return 0.0;
  }
  return eigenVec.ele[0];
  // Eigen values are sorted by ascending order.
  break;
case DenseMatrix::COMPLETION:
  rError("DenseMatrix:: no support for COMPLETION");
```

品質保証: MBLASのデバッグ

本質的にdebugが難しい

- **いろんな値を代入し、MBLASとBLASを呼んでほぼ同じ値が出るかのみチェック。**
- **アルゴリズムのチェック;BLASは代数的手法のみ用いる。**

```
for (int k = MIN_K; k < MAX_K; k++) {
 for (int n = MIN_N; n < MAX_N; n++) {
  for (int m = MIN_M; m < MAX_M; m++) {
...
   for (int lda = minlda; lda < MAX_LDA; lda++) {
    for (int ldb = minldb; ldb < MAX_LDB; ldb++) {
     for (int ldc = max(1, m); ldc < MAX_LDC; ldc++) {

      Rgemm(transa, transb, m, n, k, alpha, A, lda, B, ldb, beta, C, ldc);
      dgemm_f77(transa, transb, &m, &n, &k, &alphad, Ad, &lda,
                                            Bd, &ldb, &betad, Cd, &ldc);
...
    diff = vec_diff(C, Cd, MAT_A(ldc, n), 1);
    if (fabs(diff) > EPSILON) {
```

品質保証: MLAPACK のデバッグ

本質的に debug **が難しい**

- **ランダムに値を代入し、**MLAPACK **と** LAPACK **を呼んでほぼ同じ値が出るかチェック。**
- LAPACK **では、収束の概念が入ってくる。**
  - **代数演算のみの** BLAS **と本質的に違う。**
  - Rlamch **のパラメータがよくわからない。**Safe minimum, relative machine epsilon **の求めかたなどは妥当か。**
  - iMlaenv (ilaenv) **ブロックサイズは意味があるかどうか。**
- **研究をするとバグがとれることがある** (Waki *et al.*)

汎用性の確保

- **必要なのは** INTEGER, REAL, COMPLEX, LOGICAL **の** 4 **つの型だけ。**
- typedef **で** REAL → mpf_class, qd_real, dd_real **切り替え。**
- **初等関数** (log, sin etc) **も必要**; **だが** double **くらいの精度で良い。**
- **現在二つの多倍長精度ライブラリに対応** (GMP, QD)**。**
- **多倍長精度演算ライブラリの違いを吸収する関数をつくる。**
- C++**で書くと、ほとんど**"double" **と同じ感覚でプログラム可能。**

GMP について

- GMP(GNU Multiple Precision Arithmetic Library)
- “the fastest bignum library on the planet!”
- C++インターフェースでほぼ普通の実数型 (double) の様に使える。
- IEEE754 には従ってない。NaN, Inf などの exception のハンドリングの必要性。
- MPFR を使うべきかも。
- gcc は GMP を採用している。信頼性は高いのではないか。

QD について

一番速く、一番実用的

- **擬似** 8 **倍精度と擬似** 4 **倍精度**;double **型変数の並びで表現**; Dekker, Knuth **の定理を巧みに利用。**

$$a_{qd} = (a_1, a_2, a_3, a_4), a_{dd} = (a_1, a_2)$$

- QD (Double-Double and Quad-Double Arithmetic)
- `http://www.cs.berkeley.edu/~yozo/`
- PowerPC, Sparc64 **の四倍精度は** double-double **である。**
- C++**インターフェースでほぼ普通の実数型** (double) **の様に使える。**
- API **と思い、**QD **の高速化は別途行うべし。**
- exponent **が削れてしまうため、**IEEE754 **の四倍精度比べて精度は若干落ちる。**
- NaN, Inf **などの** exception **のハンドリングの必要性。**

言語; なぜ C++か

“double” の置き換え程度で多倍長精度化可能。

- C での多倍長精度計算; プログラムの可搬性はほぼゼロ。
- 多倍長計算の量をこなせる。計算の質が変化する。
- これまでの資産活用; SDPA-GMP, -QD, -DD で実証済み。
- (gcc に期待) 本当は REAL*8 → REAL*16 程度の簡便さが欲しい。

MBLASの現在の状況

- Multiple precision arithmetic BLAS; BLASの多倍長精度計算版。
- C++を使って書かれている。
- 76ルーチンがよくテストされている。
- インターフェースはほぼBLASと同じ。
- GMP/QD/DDソース統合。
- OpenMPでの高速化をしようとしている。
- BLAS wrapperの作成をしようとしている。

MLAPACK 現在の状況

LAPACK 3.1.1 をベース

- バージョン 0.6.0 (2009/11/24)
- (すべて)668 個コンパイルは通る; 2009/11/24 (CVS)
- 569 個コンパイルは通る; 2009/11/7
- 461 個コンパイルは通る; 2009/10/9
- 355 個コンパイルは通る; 2009/2/24
- 50 個弱が動く。
- GMP/QD/DD ソース統合。
- SDPA-GMP/-QD/-DD の MPACK サポート
- Rsyev.cpp, Rsterf.cpp: 対角化
- Rtrtri.cpp: 逆行列
- Rpotrf.cpp: コレスキー分解

MBLAS の作成方法

- Netlib からルーチンを持ってくる。(たとえば http://netlib.org/blas/drotg.f)
- 基本は手で書き換え; 小文字へ、GO TO を適当に変える、など。
- ループのアンローリングはしない。
- FORTRAN77 のコードを見て手作業で変換
- いくつかほとんど使われてないルーチンは無視した。
- 2007 年くらいから始めた。

とても時間がかかった; MLAPACK で改善へ

MLAPACK の作成方法

- Netlib からルーチンを持ってくる。たとえば dsyev.f など
- f2c という FORTRAN から C 変換するプログラムに多少手を入れて C (C++) に変換。
- sed を多く使うと人間がなんとか読めるソースになった。
- ファイルのサイズ順にソート。
- 最終的には手作業変換をした。
  - コンパイルのみ 5-20 ルーチン/day
  - バグ取りのみ 品質確認 0.3-2 ルーチン/day
  - 2009/11/24 MPACK 0.6.0 で全ルーチン 668 個の FORTRAN→C++変換は終わった。

モチベーション維持と体力勝負

MLAPACK の作成方法; sed スクリプト

```
sed -i bak -e '1,11d'
           -e ``s/d\([a-z0-9][a-z0-9][a-z0-9][a-z0-9][a-z0-9][a-z0-9][a-z0-9]\)_(/R\1(/g'' \
           -e ``s/d\([a-z0-9][a-z0-9][a-z0-9][a-z0-9][a-z0-9][a-z0-9]\)_(/R\1(/g'' \
           -e ``s/d\([a-z0-9][a-z0-9][a-z0-9][a-z0-9][a-z0-9]\)_(/R\1(/g'' \
           -e ``s/d\([a-z0-9][a-z0-9][a-z0-9][a-z0-9]\)_(/R\1(/g'' \
           -e ``s/d\([a-z0-9][a-z0-9][a-z0-9]\)_(/R\1(/g'' \
           -e ``s/d\([a-z0-9][a-z0-9]\)_(/R\1(/g'' \
           -e ``s/d\([a-z0-9]\)_(/R\1(/g'' \
           -e ``s/z\([a-z0-9][a-z0-9][a-z0-9][a-z0-9][a-z0-9][a-z0-9][a-z0-9]\)_(/C\1(/g'' \
           -e ``s/z\([a-z0-9][a-z0-9][a-z0-9][a-z0-9][a-z0-9][a-z0-9]\)_(/C\1(/g'' \
           -e ``s/z\([a-z0-9][a-z0-9][a-z0-9][a-z0-9][a-z0-9]\)_(/C\1(/g'' \
           -e ``s/z\([a-z0-9][a-z0-9][a-z0-9][a-z0-9]\)_(/C\1(/g'' \
           -e ``s/z\([a-z0-9][a-z0-9][a-z0-9]\)_(/C\1(/g'' \
           -e ``s/z\([a-z0-9][a-z0-9]\)_(/C\1(/g'' \
           -e ``s/z\([a-z0-9]\)_(/C\1(/g'' \
           -e 's/lsame_/Mlsame/g'                  \
           -e 's/integer/int/g'                    \
           -e 's/return 0/return/g'                    \
           -e 's/doublereal/mpf_class/g'               \
           -e 's/doublecomplex/mpc_class/g'  \
           -e 's/extern double dlamch_(char \*, ftnlen)//g'  \
           -e ``s/, ftnlen [a-z]*_len//g'' \
           -e 's/, (ftnlen) [0-9]//g'                  \
           -e 's/, (ftnlen) [0-9][0-9]//g'                     \
           -e 's/, ftnlen//g'                    \
           -e 's/d_sign/Msign/g'                   \
...
```

MLAPACKの作成方法; “dsteqr.f + hacked f2c + sed = dsteqr.c”

```
        i1 = l;
        for (i = mm1; i >= i1; i--) {
            f = s * e[i];
            b = c * e[i];
            Rlartg(&g, &f, &c, &s, &r);
            if (i != m - 1) {
                e[i + 1] = r;
            }
            g = d[i + 1] - p;
            r = (d[i] - g) * s + c * Two * b;
            p = s * r;
            d[i + 1] = g + p;
            g = c * r - b;

/*           If eigenvectors are desired, then save rotations. */

            if (icompz > 0) {
                work[i] = c;
                work[n - 1 + i] = -s;
            }


        }

/*        If eigenvectors are desired, then apply saved rotations. */

        if (icompz > 0) {
            mm = m - l + 1;
            Rlasr("R", "V", "B", n, &mm, &work[l], &work[n - 1 + l], &z[l * ldz + 1], ldz);
```

どれくらい速い/遅いか

- MPACK 自体を真面目に計測したことはない。
- SDPA-GMP/-QD/-DD では、100~1000 倍低速。ただし他の SDP ソルバと比較すると SDPA-DD のほうが速いこともある。
- DD(擬似四倍精度) は double と比較して 10 倍程度が最終目標。
  - 姫野龍太郎先生「10 倍くらいなら使える」
  - 後藤和茂先生「律速はデータ転送速度になるだろう」
  - Prof. Hans D. Mittelmann 「Kissing number を求めるのに SDPA-DD で 10 週間かかった」
  - ハードウェア実装が正しいか、は、微妙。

MPACK による SDPA-GMP, -QD, -DD の結果

TSPbays29.dat-s (巡回セールスパーソン問題の SDP 緩和) double だと答えが出ない。
SDPA-GMP

```
64 1.1e-24 2.9e-35 3.3e-59 +2.00e+03 +2.00e+03 7.1e-01 6.6e-01 1.00e-01
65 4.5e-25 2.9e-35 2.5e-59 +2.00e+03 +2.00e+03 7.1e-01 6.6e-01 1.00e-01

phase.value = pdOPT
   Iteration = 65
          mu = 4.4571618160501371e-25
relative gap = 3.4548724778912391e-80
         gap = 6.1785177094087001e-21
      digits = 2.6329067911852388e+02
objValPrimal = 1.9997655161769048e+03
objValDual   = 1.9997655161769048e+03
p.feas.error = 2.9085455051075594e-31
d.feas.error = 4.9280060832703393e-55
relative eps = 8.6361685550944446e-78
total time   = 525688.570
```

(146 時間=6 日)

MPACK による SDPA-GMP, -QD, -DD の結果

ハバードモデルの例 ($N = 8, L = 8, S = 0$, $U/t = 100000.0$, $PQGT1T2'$) double **だと答えが出ない** SDPA-GMP

```
63 1.0e-27 4.6e-35 3.7e-50 -2.27e-04 -2.27e-04 9.2e-01 9.5e-01 1.00e-01
64 1.4e-28 4.6e-35 1.7e-51 -2.27e-04 -2.27e-04 1.8e-01 1.8e-01 1.00e-01
65 1.2e-28 4.6e-35 1.4e-51 -2.27e-04 -2.27e-04 1.8e-01 1.8e-01 1.00e-01

phase.value = pdOPT
   Iteration = 65
          mu = 1.2108885349899566e-28
relative gap = 1.68736037617717795e-60
         gap = 3.8482037641980820e-25
      digits = 5.6128358347722269e+01
objValPrimal = -2.2675986731298406e-04
objValDual   = -2.2675986731298406e-04
p.feas.error = 4.6325094686566161e-31
d.feas.error = 2.3771386665899760e-46
relative eps = 1.2154326714572501e-63
total time   = 754492.670
```

(8.7 **日**) double **だと**

```
objValPrimal = -1.9739677666062296e-02
objValDual   = -1.9493709034577478e-02
```

今後の予定

- MLAPACK の動く関数を増やす。
- 精度の検証; デバグ
- 高速化 (GPGPU[椋木, 高橋@HOKKE 2009/12], QD/DD, マルチコア...)
- BLAS/LAPACK 互換インターフェースの作成。
- 他プラットフォームのサポート。
- ドキュメント整備。
- BLAS/LAPACK とのベンチマーク。
- MPI での massively parallel (BLACS, ScaLAPACK)
- C の long double/FORTRAN の REAL*16 を DD に (gcc に期待)!
- LAPACK 開発チームと議論して統合。