付 録A　LAPACKについて

A.1　LAPACKとは？

LAPACK の構成は以下の通り。

`LAPACK(Linear Algebra PACKage) - BLAS(Basic Linear Algebra Subprograms)`

BLAS が LAPACK の下請け的な部分を受け持つ格好となっている[1]。どちらも Fortran77 で coding されたライブラリだが，近年は C から呼び出せる CLAPACK 及び CBLAS が用意されている[2]

A.1.1　BLAS

BLAS は次の 3 つに分類される [3]。

Level 1 BLAS $y \longleftarrow \alpha x + y$（ベクトル同士の演算）

Level 2 BLAS $y \longleftarrow \alpha Ax + \beta y$ (行列–ベクトル演算)

Level 3 BLAS $C \longleftarrow \alpha AB + \beta C$ (行列同士の演算

それぞれの subroutine に対して単精度なら `S...`, 倍精度なら `D...`，単精度複素数は `C...`，倍精度複素数は `Z...` のように決まった文字が最初に来る。行列を使う演算の場合は，行列の型がを表わす文字 (列) がその次に来る。サポートしている行列型は次の通り。

一般行列 (GEneral) — `GE`

一般帯行列 (General Band) — `GB`

---

[1]「BLAS は厳密には LAPACK の一部とは言えない」[3] そうだから，こういう言い方はあまり正しくないのかな？

[2]CLAPACK の方は未だに Fotran77 の subroutine を C から呼び出せるようにしてあるだけのようである (Version 3.0 現在)。CBLAS の方はもう少し使いやすいヘッダファイルが用意されている。

対称行列 (SYmmetric) — SY

対称帯行列 (Symmetric Band) — SB

対称圧縮[3]行列 (Symmetric Packed) – SP

エルミート行列 (HErmitian) – HE

エルミート帯行列 (Hermitian Band) – HB

エルミート圧縮行列 (Hermitian Packed) – HP

三角行列 (TRiangular) – TR

三角帯行列 (Triangular Band) – TB

三角圧縮行列 (Triangular Packed) – TP

A.1.2 LAPACK

LAPACK のルーチン群は大まかに次の 3 つに分類される [3]。

ドライバルーチン (driver routine) 問題を解くために使用されるルーチン群。具体的には次の問題解決のための関数が用意されている。

連立一次方程式 $AX = B$ を満足する X を求める。

線型最小二乗問題 $\min_x \|b - Ax\|$ を満足する x を求める。

一般線型最小二乗問題

固有値問題及び一般化固有値問題

特異値問題及び一般化特異値問題

計算ルーチン (computational routine) 例えば，連立一次方程式を解く際に使う LU 分解や，固有値問題を解く際に使う QR 分解を行うルーチン群。

補助ルーチン (auxiliary BLAS にないルーチンや，その他計算に必要なルーチン群。

A.1.3 LAPACK/CLAPACK のコンパイル

Linux 環境下では，CLAPACK 及び LAPACK のコンパイルは概ね次のように進む [4]。

1. `lapack.tgz/clapack.tgz` を download し解凍する
2. `INSTALL/make.inc.LINUX` を `make.inc` に copy
3. `cd BLAS; make; -> blas_LINUX.a` が出来る
4. `cd LAPACK; make; -> lapack_LINUX.a` が出来る
5. [lapack]`g77 testprog.f lapack_LINUX.a blas_LINUX.a` として使用する
6. [clapack]`clapack.h` を適当なディレクトリに放り込む
7. [clapack]`gcc testprog.c lapack_LINUX.a blas_LINUX.a` として使用する

CLAPACK でも LAPACK と共通のライブラリを使うので，`*.a` ファイルはどちらか一通り存在すれば事足りる。

A.1.4 連立一次方程式を解く

付録の Fortran プログラムを比較して著しく違うのは，行列の要素順である。即ち

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \tag{A.1}$$

という行列は，CLAPACK の場合，次のように格納されなければならない。

$$\begin{aligned} *(da + 0) &= a_{11} \\ *(da + 1) &= a_{21} \\ *(da + 2) &= a_{31} \\ *(da + 3) &= a_{12} \\ *(da + 4) &= a_{22} \\ *(da + 5) &= a_{32} \\ *(da + 6) &= a_{13} \\ *(da + 6) &= a_{23} \\ *(da + 6) &= a_{33} \end{aligned} \tag{A.2}$$

```
/* test_dgesv.c : Tomonori Kouya         */
/* Test Program for CLAPACK 3.0 + Update */
/*                                       */
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

#include "f2c.h"
#include "clapack.h"

#define MIN(i,j) (((i) < (j)) ? (i) : (j))

main()
{
        int i, j;
        doublereal *da, *db, *dx;

        integer info, *ipiv, n, dim, lcb;

        n = 100; dim = 110; lcb = 1;

        da    = (doublereal *)calloc(sizeof(doublereal), dim * dim);
        db    = (doublereal *)calloc(sizeof(doublereal), dim);
        dx    = (doublereal *)calloc(sizeof(doublereal), dim);
        ipiv  = (integer *)calloc(sizeof(integer), dim);

        for(i = 0; i < n; i++)
                *(dx + i) = 1.0;

        for(i = 0; i < n; i++)
        {
                *(db + i) = 0.0;
                for(j = 0; j < n; j++)
                {
                        *(da + j * dim + i) = rand();
                        *(db + i) += (*(da + j * dim + i)) * (*(dx + j));
                }
                for(j = n; j < dim; j++)
                        *(da + j * dim + i) = 0.0;
        }


        for(i = 0; i < n; i++)
                printf("%5d %25.17e %25.17e\n", i + 1, *(db + i), *(dx + i));
```

```
        dgesv_(&n, &lcb, da, &dim, ipiv, db, &dim, &info);

        printf("info = %d\n", info);

        for(i = 0; i < n; i++)
                printf("%5d %25.17e %25.17e %10.3e\n", i + 1, *(db + i), \
                *(dx + i), fabs(*(db + i) - *(dx + i)) );

        free(da);
        free(db);
        free(dx);
        free(ipiv);
}
```

A.1.5 実正方行列の固有値・固有ベクトルを求める

```
/* test_dgeev.c : Tomonori Kouya         */
/* Test Program for CLAPACK 3.0 + Update */
/*                                       */
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

#include "f2c.h"
#include "clapack.h"

#define MIN(i,j) (((i) < (j)) ? (i) : (j))

main()
{
        int i, j;
        doublereal *da, *dwr, *dwi, *dvl, *dvr, *dwork;

        integer info, n, dim, n3, dim3;

        n = 10; dim = 11; n3 = n * 3; dim3 = dim * 3;

        da    = (doublereal *)calloc(sizeof(doublereal), dim * dim);
        dwr   = (doublereal *)calloc(sizeof(doublereal), dim);
        dwi   = (doublereal *)calloc(sizeof(doublereal), dim);
        dvl   = (doublereal *)calloc(sizeof(doublereal), dim * dim);
        dvr   = (doublereal *)calloc(sizeof(doublereal), dim * dim);
```

```
        dwork = (doublereal *)calloc(sizeof(doublereal), dim3);

        for(i = 0; i < n; i++)
        {
                for(j = 0; j < n; j++)
                        *(da + j * dim + i) = rand();
                for(j = n; j < dim; j++)
                        *(da + j * dim + i) = 0.0;
        }

        dgeev_("N", "N", &n, da, &dim, dwr, dwi, dvl, &n, dvr, &n, dwork, &n3, &info)

        for(i = 0; i < n; i++)
                printf("%5d %15.7e %15.7e\n", i + 1, *(dwr + i), *(dwi + i));

        free(da);
        free(dwr);
        free(dwi);
        free(dvr);
        free(dvl);
        free(dwork);
}
```

A.2　パフォーマンス計測

以下の 2 つの環境下で測定した。最適化は特に行っていない。

LINUX-PII

CPU Intel Pentium II 266MHz

RAM 128MB

OS RedHat Linux 7.0J(Kernel Version: 2.2.16-22)

Compiler gcc 2.96

LINUX-PIII

CPU Intel Pentium III 800MHz

RAM 128MB

OS RedHat Linux 7.1J(Kernel Version: 2.4.2-2)

Compiler gcc 2.96

連立一次方程式の場合は次の通り。単位は全て秒である。計測は time コマンドを使用し，user 値を採用した。

n	256	512	1024	2048
LINUX-PII	0.290	1.880	17.480	198.790
LINUX-PIII	0.080	0.740	9.850	87.180

固有値問題の場合は次の通り (固有値のみ計算)。但し，固有値問題は行列の性質に依存して収束の状態が変化するので，単純な比較は出来ないことに留意すること。

n	128	256	512	1024	2048
LINUX-PII	0.350	2.750	31.990		
LINUX-PIII	0.100	1.190	11.460	91.720	757.280

実行プログラム全体の計測結果なので，CLAPACK の関数 (dgesv, dgeev) の実行時間は幾分これよりは少ないと予想される。なお時間計測については付録で述べているように，誤差があるのでその分は増減があることを知っておく必要があろう。

A.3 g77 用のプログラム例

連立一次方程式を解くプログラムの Fortran バージョン。test_dgesv.c を同じ計算を行っている。

```
c test_tgesv.f : Tomonori Kouya
c Test Program for Lapack 3.0u
c
        integer n, i, j, dim, lcb
        parameter(n = 100, dim = 110, lcb = 1)
c
        real*8 da (dim,dim), db (dim), dx (dim)
c
        integer info, ipiv(dim)
c
        do i = 1, n
                dx(i) = 1.0
        end do
c
```

```
        do i = 1, n
                db(i) = 0.0
                do j = 1, n
                        da(i,j) = rand()
                        db(i) = db(i) + da(i,j) * dx(j)
                end do
                do j = n + 1, dim
                        da(i,j) = 0.0
                end do
        end do
c
        do i = 1, n
                write(6,600) i, db(i), dx(i)
        end do
c
        call dgesv(n, lcb, da, dim, ipiv, db, dim, info)
c
        write(6,*) 'info = ', info
c
        do i = 1, n
                write(6,600) i, db(i), dx(i), abs(db(i) - dx(i))
        end do
c
 600    format(i5, 2e25.17, e10.3)
c
        stop
        end
```

実正方行列の固有値・固有ベクトルを求めるプログラム。

```
c test_dgeev.f : Tomonori Kouya
c Test Program for Lapack 3.0u
c
        integer n, i, j, dim
        parameter(n = 10, n3=3*n, dim=11, dim3=3*dim)
c
        real*8 da(dim,dim), dwr(dim), dwi(dim), dvl(dim, dim), dvr(dim, dim), dwork(di
c
        integer info
c
        do i = 1, n
                do j = 1, n
                        da(i,j) = rand()
                end do
                do j=n+1,dim
```

```
                        da(i,j) = 0.0
                end do
        end do
c
        call dgeev('N', 'N', n, da, dim, dwr, dwi, dvl, n, dvr, n, dwork, n3, info)
c
        do i = 1, n
                write(6,600) i, dwr(i), dwi(i)
        end do
c
600     format(i15, (4e15.8))
c
        stop
        end
```