ハイパフォーマンスコンピューティング：ライブラリの利用

線形計算ライブラリ

- 線形問題を数値計算で解くためのライブラリ。非常に高速かつ信頼性の高い関数。また、移植性が高く、幅広いコンピュータで使う事ができる。

  - BLAS・・・Basic Linear Algebra Subprograms。基本的な線形計算を行うサブルーチンを集めたもの。
    - BLASの関数仕様は大きく3種類
      - LEVEL1 BLAS　ベクトル演算（例：$y \leftarrow \alpha x+y$）
      - LEVEL2 BLAS　行列ーベクトル演算（例：$y \leftarrow \alpha Ax+\beta y$）
      - LEVEL3 BLAS　行列演算（例：$C \leftarrow \alpha AB+\beta C$）

  - LAPACK・・・Linear Algebra PACKage。線形代数問題（連立一次方程式や固有値問題等）を解くことができる。BLASを利用。

  - ScaLAPACK ・・・分散メモリ型のコンピュータ向けに開発された高性能なルーチン群。BLASやLAPACK、MPI等を利用。

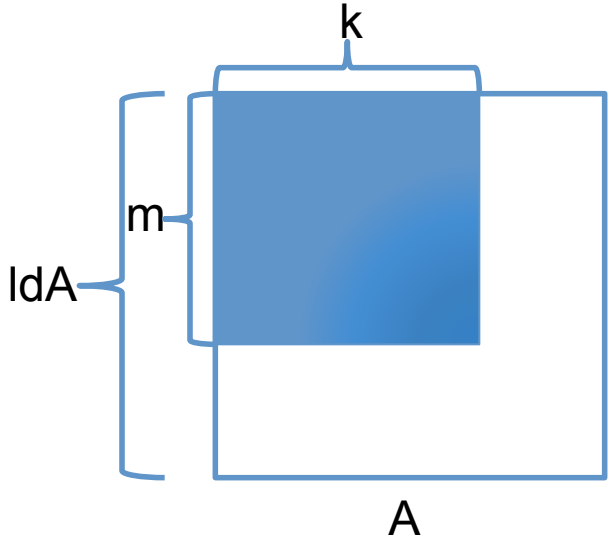
  - ※Netlib：http://www.netlib.org/liblist.htmlからダウンロード出来ます。

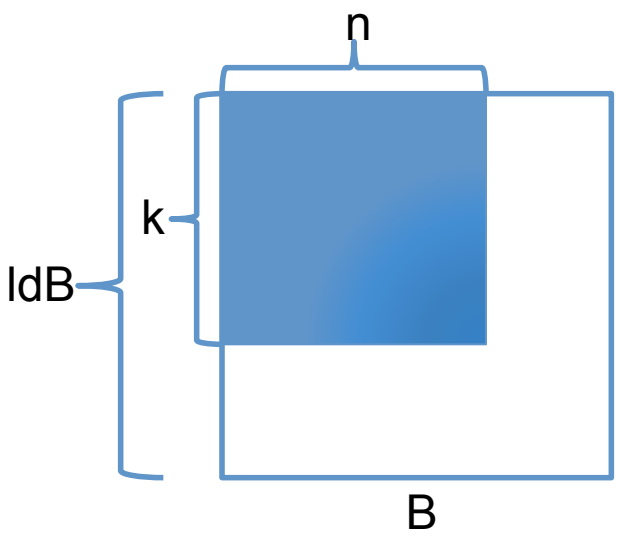
BLAS：行列積

- Xgemm :行列演算を実行するルーチン
  - Xはどれか1つ: s: float, d: double, c: complex, z: double complex
  - C := α*op(A)*op(B) + β*C

【呼び出し方】

- dgemm (transA, transB, m, n, k, α, A, ldA, B, ldB, β, C, ldC)
  - transA, transB: ‘N’（転置にしない）, ‘T’（転置にする）, ‘C’（共役転置にする）が入る





BLASのインストール

1. NetlibのサイトからBLAS（blas.tgz）をダウンロードする。
2. 解凍（BLASディレクトリが作成される）

```
$ tar xzvf blas.tgz
$ cd BLAS
```

3. make（gfortran等のFortranコンパイラが必要）。
   - BLASディレクトリにある make.inc を編集し、Fortranコンパイラを設定することが可能。

```
$ make
```

4. make後、blas_LINUX.aがあることを ls 等で確認。
5. blas_LINUX.a を libblas.a のようにリネーム（必須ではない）。

```
$ mv  blas_LINUX.a  libblas.a
```

6. libblas.a を /usr/local/lib などに移動（管理者権限がないと不可）。

※ 最適化BLASであるGotoBLASとATLASは無償でソースコードを配布しています。これを入手してmakeすれば使用可能になります。

※ Mac OS Xには、Intel-Core CPU用に最適化されたBLASを含んだライブラリ、vecLibが用意されているので、それを使うことも可能。

BLASライブラリの呼び出し（リンク方法）

- libblas.a が /home/user/lib に格納されている場合

  $ gcc -O2 foo.c /home/user/lib/libblas.a -lgfortran

  （gfortranで make した場合は, -lgfortran が必要）

  あるいは

  $ gcc -O2 foo.c -L/home/user/lib -lblas -lgfortran

- libblas.a が /usr/local/lib 等に格納されている場合

  $ gcc -O2 foo.c -lblas -lgfortran

※ Mac OS X (vecLib)の場合

$ gcc -O2 foo.c -framework vecLib

演習（BLAS）

- BLASを呼び出して行列積を計算し、どれだけ高速化されるか確認しよう。
- leading dimensionを利用した1次元配列による行列の表現のソースを応用しよう。
- サンプルファイル「t_blas1.c」が授業のWebページにあるので、それを参考にしよう。

ヒント

・自分で作った行列積関数multmm（C←A×B）を呼び出す場合

```
multmm(m,p,n,A,m,B,p,C,n);
```

・BLASの行列積ルーチンを呼び出す記述

```
char  trans = 'N';
double  one = 1.0,  zero = 0.0;
dgemm_(&trans, &trans, &m, &n, &p, &one, A, &m, B, &p, &zero, C, &n);
```

※ ルーチン名の最後にアンダースコア_が付いていることに注意してください。
※ C言語からFortranのルーチンを呼び出す場合、引数はアドレス渡しになります。

LAPACKのインストール

1. BLASと同様にNetlibからLAPACKのソースファイルをダウンロード＆解凍
2. INSTALLディレクトリから適切なmake.incファイルをコピーする。
   - 例えば、gfortranを利用するなら

   $ cd (LAPACKの解凍先ディレクトリ)

   $ cp ./INSTALL/make.inc.gfortran make.inc
3. make.incを編集し、BLASLIBの行を適切なものに修正

   BLASLIB   = /home/user/lib/libblas.a
4. makeを実行

   $ make
5. ライブラリ（liblapack.a）ができていることを確認

LAPACKライブラリの呼び出し（リンク方法）

- libblas.aとliblapack.a が /home/user/lib に格納されている場合（ライブラリの順番に注意）

  $ gcc -O2 foo.c -L/home/user/lib -llapack -lblas -lgfortran

- libblas.aとliblapack.a が /usr/local/lib 等に格納されている場合は，-Lオプションは不要

  $ gcc -O2 foo.c -llapack -lblas -lgfortran

※ Mac OS X (vecLib)の場合は，BLASのときと同様

$ gcc -O2 foo.c -framework vecLib

演習（LAPACK）

- LAPACKの関数を試しに使ってみよう。
- サンプルファイル「t_lapack1.c」，「t_lapack2.c」が授業のWebページにあるので、それを参考にしてみよう。
  - t_lapack1.c: 連立一次方程式を解く（LAPACK/DGESV）
  - t_lapack2.c: 実対称固有値問題を解く（LAPACK/DSYEV）

- それぞれの関数については，LAPACKのドキュメントを読んで使用方法を理解しておこう。