ABCLibDRSSED (リリース版α) 利用の手引

片桐 孝洋[†,††]

† 電気通信大学大学院情報システム学研究科情報ネットワーク学専攻
†† 科学技術振興事業団 戦略的創造研究推進事業
さきがけプログラム「情報基盤と利用環境」領域
E-mail：katagiri@is.uec.ac.jp

平成 15 年 3 月 24 日
平成 15 年 5 月 27 日 改訂

まえがき

本マニュアルは，自動ブロック化・通信最適化ライブラリ ABC-LIB （Automatically Blocking and Communication-adjustment Library）プロジェクト [1] において開発された，密対称実数行列に関する標準固有値問題を解くためのライブラリ ABCLibDRSSED (Dense Real Symmetric Standard Eigenvalue Decomposition) の使用方法を説明したものです．

ABC-LIB は，階層型メモリを要素計算機（Processing Element, PE）が有する並列計算機上で高性能を達成する線形数値計算ライブラリです．具体的に ABC-LIB は，安価な PC を PE としてネットワーク網で繋げた PC クラスタはもちろんのこと，スーパーコンピュータ上でも高い性能を発揮するように設計された並列数値計算ライブラリです．

ABC-LIB プロジェクトでは，以下に示す方針でライブラリ開発を進めています．

- 階層型メモリを有する計算機上で高性能な数値計算を実現するため，ブロック化アルゴリズムを積極的に採用します．
- 低速なネットワーク環境でも高性能を達成するため，通信隠蔽アルゴリズムを積極的に採用します．
- PC クラスタからスーパーコンピュータに至る幅広い並列実行環境に対応するため，自動チューニング機能を搭載します．
- 自動チューニング機能の搭載により，ユーザーが指定するパラメタが従来のライブラリに比べ格段に少なく，利用がしやすいです．さらに性質上，パラメタが少ないことによる性能低下が生じることがありません．

ABC-LIB が提供する線形計算機能は，以下の機能を準備する予定です．

- 密行列用
  - 連立一次方程式の直接解法
  - 固有値問題における直接解法

  - 固有値問題における反復解法

- 疎行列用
  - 連立一次方程式の直接解法
  - 連立一次方程式の反復解法
  - 固有値問題における反復解法

本ライブラリは以下のような特長をもっています.

- インストール時，実行起動前，実行時のパラメタ最適化の枠組により，性能に関連するパラメタを自動調整 (自動チューニング) します.

上記で述べた自動チューニングソフトウエア構成方式を，FIBER(Framework of Installation, Before-invocation, and Run-time optimization layers，ファイバー) [2, 3, 4]* と呼びます．FIBER ソフトウエアアーキテクチャの機能の一部を実装したテストタイプとして，ABCLibDRSSED は開発されました.

ABCLibDRSSED はサブルーチン形式で使用します．FORTRAN 言語から呼び出すことが可能です.

ABCLibDRSSED リリース版αは，以下の処理に関して並列処理を行うことができます.

- 密対称実数行列について，相似変換による三重対角化
- 密対称実数行列について，任意の個数 (1 個から全部) の固有値計算
- 密対称実数行列について，任意の個数 (1 個から全部) の固有値および固有ベクトル計算
- エルミート行列について，全固有値計算
- エルミート行列について，全固有値および全固有ベクトル計算
- 複数の実数ベクトルについて，それら全てを直交化する処理

なお本リリース版αは，多くのユーザに利用してもらうことで，ユーザビリティの改善，バグの発見と修復，および新規開発事項への反映，を目的にリリースされるものです.

したがってマニュアルにおいての説明不備，低いコードのリーダビリティ，および突然の仕様変更，などの問題が生じる可能性があることをご理解ください.

また何かお気づきの点がありましたら，お気軽に著作者までご一報くださいますようお願い申し上げます.

* 特許出願申請済 (特願２００３－０２２７９２，平成１５年1月３０日) です．また FIBER ソフトウエアアーキテクチャをもちいた自動チューニングソフトウエアを容易に実現するための専用言語ABCLibScript[5] を開発中です．ABCLibScript に関する技術は，特許出願済です [6].

1 はじめに

本ライブラリABCLibDRSSEDは，密対称実数標準固有値問題を解くために用いることができます.

密対称実数標準固有値問題を解くとは，係数行列 $A \in \Re^{n \times n}$，ここで $A = A^T$，全固有ベクトルを列ベクトルにもつ行列 $X \in \Re^{n \times n}$，全固有値を対角要素にもつ行列 $\Lambda \in \Re^{n \times n}$ としたとき，固有分解 $A = X\Lambda X$ を行うことです.

また密対称実数標準固有値問題の解法を利用することで，エルミート行列 $A \in \Im^{n \times n}$，ここで $A = A^H$，の全固有値と全固有ベクトルも求めることができます.

本ライブラリは，Householder 法を利用した三重対角化を用いています．固有分解を完全に行うには，得られた三重対角行列 T に対して全固有値を二分法，全固有ベクトルを逆反復法で求め， Householder 変換時に得られるスカラー，ベクトルを用いて全固有ベクトルを Householder 逆変換します.

本ライブラリのマルチプロセッサ版は，MPI(メッセージ・パッシング・インタフェース)-1 を用いているため，MPI-1 が実装されている計算機上で動作します.

1.1 提供されるライブラリ (サブルーチン) 一覧

以下に，ABCLibDRSSED リリース版αが提供する機能を説明します.

- 密対称実数行列用三重対角化ルーチン [一般版] (ABCLibTriRed)：
  相似変換により三重対角化された行列が得られます．また，相似変換の際必要となる行列 H の情報も得られます.

- 密対称実数行列用三重対角化ルーチン [特化版] (ABCLibTriRedR)：
  相似変換により三重対角化された行列が得られます．なお，相似変換の際必要となる行列 H の情報は，このルーチンでは得られません．しかし実行時間は，ABCLibTriRed ルーチンよりも高速となります.

- 密対称実数行列用固有値求解ルーチン (ABCLibDRSAllEig)：
  密対称実数行列における，任意の個数の固有値を計算します.
  　なお本ルーチンでは，絶対値の最も大きい固有値から連続して m 個求めることができます.

- 密対称実数行列用固有値および固有ベクトル求解ルーチン (ABCLibDRSAllEigVec)：
  密対称実数行列における，任意の個数の固有値と固有ベクトルを計算します.
  　なお本ルーチンでは，固有ベクトルに関して対応する固有値を絶対値の大きい順にソートし番号を付けたとき，連続する m 個の固有値と固有ベクトルを求めることができます．この指定には，ソート後番号づけられた固有値についての開始番号と終了番号を与えます.

- エルミート行列用全固有値求解ルーチン (ABCLibHerAllEig)：
  エルミート行列における，全固有値を計算します.

- エルミート行列用全固有値および全固有ベクトル求解ルーチン (ABCLibHerAllEigVec)：
  エルミート行列における，全固有値と全固有ベクトルを計算します.

- Gram-Schmidt 直交化ルーチン (ABCLibQRD)：
  入力した複数の実数ベクトルにおいて，直交化を行います.

1.2 Householder 相似変換

本ライブラリ ABCLibDRSSED で利用するアルゴリズムは，以下に示す良く知られている Householder 相似変換を用いています.

[**定理**] ベクトル $x \in \Re^n$ が与えられるとすると，以下に示すベクトル $u \in \Re^n$ とスカラ $\alpha \in \Re$ が存在します： $(I - \alpha uuT)x = (v_1, ..., v_k, \sigma, 0..., 0)^T$, ここで $\sigma = ||x_{k+1:n}||_2$. □

ベクトル $u = (0, ..., 0, v_{k+1} \pm \sigma, v_{k+2}, ..., v_n)^T$, とスカラ $\alpha = 1/(\sigma^2 + |v_{k+1}\sigma|)$ は 上記の定理を満たします．また $||u||_2^2 = (v_{k+1} + = \sigma)^2 + v_{k+2}^2 + ... + v_n^2 = \sigma^2 + \sigma^2 + 2|v_{k+1}\sigma| = 2(\sigma^2 + |v_{k+1}\sigma|) = 2/\alpha$ なので $\alpha u^T u = 2$ である．σ の符号はベクトル u の計算時の桁落ちを防ぐ為に v_{k+1} と同符号にとります．ここで変換 $(I - \alpha uu^T)x$ を $H(k)(x)$ と表記します． この変換は要素 $v_1, ..., v_k$ に作用しません．さらに変換 $H(k)(x)$ に必要なベクトルとスカラの組みを (u, α) と表記します.

ここでは $A(1) = A$ から 三重対角行列 $A(n-2) = T$ への変換を考えます．$H(k) = I - \alpha uu^T$ を行列 A に関する $k+1$ 番目の反復に適用することで, 以下に示す式を得ることができます：

$$\begin{aligned} A(k+1) &= H(k)A(k)H(k) \\ &= A(k) - \alpha A(k)uu^T - \alpha uu^T A(k) - \alpha^2 uu^T A(k)uu^T \\ &= A(k) - xu^T - uy^T + \alpha uu^T xu^T \\ &= A(k) - uy^T + u\mu uT - xu^T = A(k) - u(y^T - \mu u^T) - xu^T \end{aligned} \quad (1)$$

ここで $x = \alpha A(k)u$, $y^T = \alpha u^T A(k)$, $\mu = \alpha u^T x$.
いま A は対称なので $x = y$ であり，以下の式を得ます.

$$A(k+1) = A(k) - u(x^T - \mu u^T) - xu^T \quad (2)$$

ここで k 番目の反復では，行列 $A_{k:n,k:n}$ における列ベクトル $A_{k:n,k}$ が必要であることに注意します．この列ベクトル $A_{k:n,k}$ を枢軸ベクトルと呼びます.
以下に三重対角化アルゴリズムの概略をのせます.

```
c Pmyidx,myidy owns row set Pi and column set Gamma of n * n matrix A.
1: do k=1, n-2
2: if (k \in Gamma) then
3: Broadcast(A(k)_{Pi,k}) to PEs sharing rows Pi.
4: else
5: receive(A(k)_{Pi ,k})
```

```
6: endif
7: Computation of (u_Pi, alpha).
8: if (I have diagonal elements of A) then
9: Broadcast(u_Pi) to PEs sharing columns Gamma.
10: else
11: receive(u_Gamma)
12: endif
13: do j=k, n
14: if (j \in Gamma) x_Pi = x_Pi + alpha A(k)_{Pi,j} u_j endif
15: enddo
16: Global summation of x_Pi to PEs sharing rows Pi.
17: if (I have diagonal elements of A) then
18: Broadcast(x_Pi) to PEs sharing columns Gamma.
19: else
20: receive(x_Gamma)
21: endif
22: do j=k, n
23: mu = alpha u_Pi^T x_Pi enddo
24: Global summation of mu to PEs sharing rows Pi.
25: do j=k, n
26: do i=k, n
27: if (i \in Pi .and. j \in Gamma) then
28: A(k+1)_{i,j} = A(k)_{i,j} - ui (y^T_j - mu u^T_j) - y_i u^T_j
29: endif enddo enddo
c Remove k from active columns and rows.
30: if (k \in Gamma) Gamma = Gamma - { k } endif
31: if (k \in Pi) Pi = Pi - { k } endif
32: enddo
```

このアルゴリズムでは， 2: – 12: で枢軸ベクトルの転送， 13: – 15: で行列-ベクトル積 $x = \alpha A(k)u$， 22: – 24: で内積計算 $\mu = \alpha u^T x$，そして 25: – 29: で行列更新 $A(k+1) = A(k) - u(x^T - \mu u^T) - xu^T$ の各処理を行なっています．

1.3 サイクリック-サイクリック分散方式

三重対角化およびそれに基づく全固有値，全固有ベクトル計算の並列化には，行列 A の分散に関してサイクリック-サイクリック分散方式を仮定しています† ．

入力データは標準固有値問題 $Ax = \lambda x$ における行列 A の要素です．この要素をサイクリック-サイクリック分散方式で各 PE に分散させ，それを入力データとします．

サイクリックサイクリック分散方式について，以下に具体的な説明を示しま す．まず nprocs 個のプロセッサの一次元的な番号を pe_num とします．このとき

† ただしエルミート行列の解法では，データ分散をせず，逐次プログラムと同様のインタフェースを用いて引き渡します．

pe_num : 0, 1, ..., nprocs-1
の範囲をとるものとします．また，nprocs に関して
nprocs = ncelx * ncely
とし，さらに

- ncelx < ncely, このとき ncelx は 1 であってはならない,
- ncely / ncelx は割り切れる,

こととします．ここで一次元的な番号 pe_num から二次元的なプロセッサ番号 (nmyidx, nmyidy) への変換は，以下のサブプログラムで行なっています．

```
subroutine cid2xy(pe_num, ncelx, nmyidx, nmyidy)
integer pe_num, ncelx, nmyidx, nmyidy

nmyidx = modulo(pe_num, ncelx)
nmyidy = (pe_num - modulo(pe_num, ncelx))/ncelx
return
end
```

いま，二次元的なプロセッサ番号 (nmyidx, nmyidy) が得られました．さて 行列 n × n 行列 A を nprocs プロセッサに分配するには以下のように書きます (ただし実装はしやすいが，あまり実行効率は良くない方法です)．
いま行列 A の 添字は 0〜$n-1$ まで変化するとします．

```
do i=0,n-1
  li = LOC(i)
  if (nmyidx .eq. OWNER(i)) then
    do j=0,n-1
      if (nmyidy .eq. OWNER(j)) then
        A(li, LOC(j)) = n - i
      endif
    enddo
  endif
enddo
```

ここで，OWNER(i) は第 i 列を所有するプロセッサ群の二次元的な番号をもつ配列，LOC(i) は第 i 列を所有するプロセッサのローカルな添字番号をもつ配列です．このリストの作成方法は

```
do i=0, n-1
  LOC(i) = CYCLIC_LOC(i,ncelx)
  OWNER(i) = CYCLIC_OWNER(i,ncelx)
enddo
```

です．関数 CYCLIC_LOC(i,ncelx)，CYCLIC_OWNER(i,ncelx) は

```
integer function CYCLIC_OWNER(i, nprocs)
integer i, nprocs
CYCLIC_OWNER = modulo(i, nprocs)
return
end

integer function CYCLIC_LOC(i, nprocs)
integer i, nprocs
CYCLIC_LOC = idfloor(i/nprocs)
return
end
```

このような関数です．

第 i 列を所有するプロセッサの二次元的な番号の求め方が分かれば，第 j 列を有するプロセッサの二次元的な番号も同様なので，列方向に関する所有情報は求める必要はありません．

具体的に書くと (nprocs=4 ,n=3)
行列 A =(aij),i,j=1,...,3
↓ (i) → (j)
a11, a12, a13
a21, a22, a23
a31, a32, a33

OWNER(i)，所有者情報 (nmyidx, nmyidy)
0 , (0,0) (0,1) (0,0)
1 , (1,0) (1,1) (1,0)
0 , (0,0) (0,1) (0,0)

LOCAL(i)，内部添字情報 (local_i, local_j)
0 , (0,0) (0,0) (0,1)
0 , (0,0) (0,0) (0,1)
1 , (1,0) (1,0) (1,1)

PE pe_num , (nmyidx, nmyidy)
0 , (0, 0)
1 , (1, 0)
2 , (0, 1)
3 , (1, 1)

となります．ここで 所有者情報は 行列 aij を所有している 2 次元的な PE 番号を意味します．また 内部添字情報は 行列 aij を代入するべきローカルな添字番号を示します．

例えば 0 番のプロセッサ $(0, 0)$ は，行列 A の要素

a11, a31, a13, a33

を所有していることが所有者情報からわかり，ローカルに代入すべき場所もそれぞれ

(0,0), (1,0), (0,1), (1,1)

に代入すべきことが 内部添字情報から分かります.

2 自動チューニング項目

この章では，本ライブラリで実装されている自動チューニングについて説明します.

2.1 FIBERによる自動チューニングの分類

FIBER ソフトウエアアーキテクチャでは，自動チューニング処理を以下に示す 3 種の処理に分類分けしています.

- インストール時自動チューニング：
  ライブラリをインストールした時に行う自動チューニング
- 実行起動前自動チューニング：
  ユーザが与えた基本情報 (プロセッサ台数，問題サイズなど) を指定した後に行う自動チューニング
- 実行時自動チューニング：
  ライブラリ実行時に行う自動チューニング

ABCLibDRSSED リリース版αでは，FIBER ソフトウエアアーキテクチャの機能の一部を実装しています．具体的には，インストール時自動チューニング機能，および実行起動前自動チューニング機能の一部を行うことができます‡ .

2.2 ライブラリ構成

本ライブラリは，主に以下の 4 つの部分に分かれて構成されています.

1. Householder 三重対角化部
2. 二分法による固有値計算部
3. 逆反復法による固有ベクトル計算部
4. Householder 逆変換による固有ベクトル変換部

‡ 現在のバージョンでは，実行起動前自動チューニングは手動で行います (5 章 4 節 「自動チューニングの実行」を参照).

現在のバージョン (リリース版α) では，1. および 4. の処理部において，自動チューニング機能を搭載しています.

また現在のバージョンの固有値および固有ベクトル計算処理には用いていないものの，QR 分解などの線形計算に利用できる Gram-Schmidt 法による直交化ルーチンが本ライブラリに含まれています．この直交化ルーチンにも，自動チューニング機能を搭載しております.

2.3 Householder 三重対角化部の自動チューニング

Householder 三重対角化部における自動チューニング項目は，以下の通りです.

- 行列-ベクトル積のループアンローリング
- 行列更新処理のループアンローリング
- 通信方式

行列-ベクトル積のループアンローリングは，行列-ベクトル積の演算ループをに関して，最内側から数えて 2 番目のループを 1,2,...,16 段展開したものをそれぞれ用意しており，この段数を自動調整します.

行列更新処理のループアンローリングは，行列更新処理の演算ループをに関して，最内側から数えて 2 番目のループを 1,2,...,16 段展開したものをそれぞれ用意しており，この段数を自動調整します.

通信方式は，三重対角化の際に必要となるベクトルに対するプロセッサ間加算処理に対して，下記の 2 つの方法から最も早く計算できる方式を自動選択します.

- 二進木構造の通信を用いて実現する方式：
  MPI ライブラリにある同期通信関数`MPI_Recv`，`MPI_Send` を使う方式.
- MPI 関数を用いる方式：
  MPI ライブラリにある`MPI_Allreduce` を使う方式.

2.4 Householder 逆変換による固有ベクトル変換部

Householder 逆変換による固有ベクトル変換部における自動チューニング項目は，以下の通りです.

- 行列更新処理のループアンローリング
- 通信方式

行列更新処理の演算ループをに関して，最内側から数えて 2 番目のループを 1,2,...,16 段展開したものをそれぞれ用意しており，この段数を自動調整します.

通信方式は，必要となるベクトルの放送処理に対して，下記の 3 つの方法から最も速く計算できる方式を自動選択します.

- MPI 関数を用いる方式：
  MPI ライブラリにあるMPI_Bcast を使う方式.
- 同期通信を用いて二進木構造の通信を用いて実現する方式：
  MPI ライブラリにある同期通信関数MPI_Recv，MPI_Send を使う方式.
- 非同期通信を用いて二進木構造の通信を用いて実現する方式：
  MPI ライブラリにある同期通信関数MPI_Recv，非同期通信関数MPI_Isend を使う方式.

2.5 Gram-Schmidt 直交化ルーチン

Gram-Schmidt 直交化ルーチンにおける自動チューニング項目は，以下の通りです.

- ブロック幅
- 枢軸 PE 用の行列更新処理のループアンローリング
- 主行列更新処理のループアンローリング

Gram-Schmidt 直交化ルーチンでは，ブロック化アルゴリズムを採用しており，このブロック幅が通信性能と演算性能に影響します．このブロック幅に関して，1,2,3,4,5,6,8,16 まで指定することができ，この幅を自動調整します.

枢軸 PE 用の行列更新処理に関して，最も外側，外側から 2 番目のループについて，それぞれ 1,2,3,4 段 および 1,2,3,4,5,6,8,16 段の，計 $4 \times 8 = 32$ 通り展開したものを用意しており，この段数を自動調整します.

主行列更新処理に関しては枢軸 PE 用の行列更新処理と同様に，最も外側，外側から 2 番目のループについて，それぞれ 1,2,3,4 段 および 1,2,3,4,5,6,8,16 段の，計 $4 \times 8 = 32$ 通り展開したものを用意しており，この段数を自動調整します.

3 インストール方法

3.1 手順

提供されているファイルabclib_r_alpha.tar.gz を，解凍して展開してください.
例えば UNIX では，以下の手順で行います.

```
% gzip -d abclib_r_alpha.tar.gz
% tar xvf  abclib_r_alpha.tar
```

上記展開後，ABCLib_Ralpha ディレクトリのなかにあるコマンドabclib_install[§] を，以下のように実行してください.

[§] なおtests/Makefileは，PGI コンパイラを利用する場合に特化されています．適宜，コンパイラコマンドおよび最適化オプションを書き換えてから実行してください.

```
% cd ABCLib_Ralpha
% ./abclib_install
```

abclib_install コマンドにより，全てのサンプルプログラム，および，静的ライブラリが生成されます．生成される静的ライブラリは，abclib_install コマンドにより生成されるディレクトリABCLib_Ralpha/lib の下にあるABCLib.aです．

3.2 ディレクトリ構造

展開後のABCLibDRSSEDリリース版αのディレクトリ構造は，以下のようになっています．

```
ABCLib_Ralpha/ --- DRSSED/ --- src/ --- *.f
               |
               |__ doc/ --- *.ps
               |         |__ *.pdf
               |
               |__ include/ --- ABCLib*.h
               |
               |__ tests/ --- Makefile
               |           |__ ABCLibTest*.f
               |
               |__ COPYRIGHT
               |__ KnownBugs
               |__ README
               |__ abclib_install
               |__ abclib_uninstall

               |__ lib -- ABCLib.a (注：abclib_install 実行後生成)
```

ディレクトリDRSSED/は，本固有値ソルバのソースコードが入っているディレクトリです．
ディレクトリdoc/は，ABCLibDRSSED関連のドキュメントが入っているディレクトリです．
ディレクトリinclude/は，ABCLib に必要なインクルードファイルが入っているディレクトリです．
ディレクトリtests/は，利用のためのサンプルプログラムが入ってが入っているディレクトリです．これらのサンプルプログラムをコンパイルすることで，利用方法の確認，およびライブラリの起動確認ができます．
ディレクトリlib/は，静的ライブラリが設置されるディレクトリです．コマンド abclib_install 実行後に生成されます．またコマンド abclib_uninstall 実行後には，消去されます．

4 サンプルプログラム

4.1 対称実数密行列の固有値，固有ベクトルの求解

対称実数密行列の固有値，固有ベクトルを求めるためのサンプルプログラム (tests/ABCLibTestDRSAllEigVec.f) は，以下のとおりです.

```
      program  main
      include  'ABCLibDRSSED.h'
      common  /ABCLibpval/ nprocs, myid, ncelx, ncely, nx, ny, ierrcode

c     === real*8 definition
c     ============================================================
      real*8  A(0:ABCLibMAXNX-1, 0:ABCLibMAXNY-1)
      real*8  eig(1:ABCLibMAXNDIVP)
      real*8  X(1:ABCLibMAXN, 1:ABCLibMAXNDIVP)

c     === for checking answer
      real*8  dtemp
      real*8  EigErr, VecErr, SysErr

c     === for measuring time
      real*8    t1, t2, bt, t_all
c     ============================================================

c     === integer definition
c     ============================================================
c     === for parallel control
      integer   ncelx, ncely, nx, ny
      integer   myid, nmyidx, nmyidy
      integer   n, m, ms, me, nprocs

c     === for error flag
      integer   ierrcode
c     === for selecting test matrices
      integer   isw_mat
c     ============================================================

c     -=- Program start
c     =-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=

c     === MPI Init.
c     ====================================================
```

```
      call MPI_INIT( ierr )
      call MPI_COMM_RANK( MPI_COMM_WORLD, myid, ierr )
      call MPI_COMM_SIZE( MPI_COMM_WORLD, nprocs, ierr )
c     ====================================================

c     === Print title
c     ====================================================
      if (myid .eq. 0) then
        print *, ""
        print *, "=== Parallel Eigensolver Check Program ==="
        print *, "Parallel Eigensolver on MPI"
        print *, "ABCLibDRSAllEigVec"
        call ABCLibPrintVer()
        print *, ""
      endif
c     ====================================================

c     === Setting Test Condition
c     ====================================================
      if (myid .eq. 0) then
        write(6,*)  "Problem size >"
        read(5,*) n
        write(6,*)  "Start number of eigenvectors >"
        read(5,*) ms
        write(6,*)  "End number of eigenvectors >"
        read(5,*) me
        m = me - ms + 1

c       === set matrix type
        isw_mat = 1
      endif
c     ====================================================

c     == Set Parameters
c     ====================================================
      call MPI_BCAST(n,1,MPI_INTEGER,0,MPI_COMM_WORLD,ierr)
      call MPI_BCAST(m,1,MPI_INTEGER,0,MPI_COMM_WORLD,ierr)
      call MPI_BCAST(ms,1,MPI_INTEGER,0,MPI_COMM_WORLD,ierr)
      call MPI_BCAST(me,1,MPI_INTEGER,0,MPI_COMM_WORLD,ierr)
      call MPI_BCAST(isw_mat,1,MPI_INTEGER,0,MPI_COMM_WORLD,ierr)
c     ====================================================
```

```
c     === Set Parallel Control Values
c     ================================================
      if (myid .eq. 0) then
        write(6,*) " ncelx="
        read(5,*) ncelx
        write(6,*) " ncely ="
        read(5,*) ncely
      endif
      call MPI_BCAST(ncelx, 1, MPI_INTEGER, 0, MPI_COMM_WORLD, ierr)
      call MPI_BCAST(ncely, 1, MPI_INTEGER, 0, MPI_COMM_WORLD, ierr)
      call cid2xy2(myid, ncelx, ncely, nmyidx, nmyidy)
      nx = ceiling(dfloat(n)/dfloat(ncelx))
      ny = ceiling(dfloat(n)/dfloat(ncely))
c     ================================================

c     === Generate Test Matrix
c     ================================================
      call MakeTestMat2(A, n, nmyidx, nmyidy, isw_mat)
c     ================================================

c     === Initialize ABCLibTriRed routine
c     ================================================
      call ABCLibDRSAllEigVec_Init()
c     ================================================

c     === Call Tridiagonalication Routine
c     ================================================
      call MPI_BARRIER(MPI_COMM_WORLD, ierr)
      t1 = MPI_WTIME()

      call ABCLibDRSAllEigVec(A, n, eig, X, ms, me)

      call MPI_BARRIER(MPI_COMM_WORLD, ierr)
      t2 = MPI_WTIME()

      t_all = t2 - t1
      call MPI_REDUCE(t_all, bt, 1, MPI_DOUBLE_PRECISION, MPI_MAX, 0,
     &                MPI_COMM_WORLD, ierr)
      t_all = bt
c     ================================================

c     === Calculate Eigenvalues & Check Its Error
```

```
c     ====================================================
      if (ierrcode .eq. 0) then
        if (myid .eq. 0) print *, "Normal return"
      else
        print *, "myid=",myid,"Abnormal return: error code",ierrcode
      endif

      if (isw_mat .eq. 1) then
        if (myid .eq. 0) print *,
     &     "Check of maximal error for eigenvalues...."
        call ChkEigErr(eig, n, EigErr, ms, me)
        if (myid .eq. 0) print *, "End of checking."
      endif

        if (myid .eq. 0) print *,
     &     "Check of error for eigenvectors...."
        call TestVec(X, n, VecErr, m)
        if (myid .eq. 0) print *, "End of checking."

      if (myid .eq. 0) print *,
     &   "Check of error for the eigensystem...."
        call TestEigSys(eig, X, n, isw_mat, SysErr, m)
      if (myid .eq. 0) print *, "End of checking."
c     ====================================================

c     === Output Results
c     ====================================================
      if (myid .eq. 0) then
        print *, ""
        print *, "=== Result "
        print *, "================================================ "
        print *, "n = ", n, " ,Number of processors = ", nprocs
        print *, "Number of eigenvalues and eigenvectors = ", m
        print *, "Start Eigenvalue No.", ms, "/ End Eigenvalue No.",me
        print *, "Test Matrix:"
        if (isw_mat .eq. 1) print *, "Frank"
        if (isw_mat .eq. 2) print *, "Rnd (-32768 ~ 32765)"
        if (isw_mat .eq. 4) print *, "Glud Wilkinson W_21, d=1.0d-14"
        if (isw_mat .eq. 5) print *, "Unit Matrix"
        if (isw_mat .eq. 6) print *, "Wilkinson W_n+"
        print *, ""
        if (isw_mat .eq. 1) then
```

```
          print *, "Eigenvalue Max Error = ", EigErr
        endif
        print *, "Eigenvector Error of ||X^T X - I|| = ", VecErr
        print *,
     &    "Eigensystem Error max_i||A x_i - lambda_i x_i|| = ", SysErr

        write(*, 1000) t_all
 1000   format(" Total Calculation Time = ",F10.3,"  [sec]")
        print *, ""
      endif
c     ====================================================

c     === Finalize ABCLibDRSAllEig routine
c     ====================================================
      call ABCLibDRSAllEigVec_Finalize()
c     ====================================================

c     ===== MPI finazize
c     ====================================================
      call MPI_FINALIZE(ierr)
c     ====================================================

c     =-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
      stop
      end
```

4.1.1 サンプルプログラムの説明

ABCLibDRSSED を利用するには，以下のヘッダーファイル，および common 文を付加してください.

```
include 'ABCLibDRSSED.h'
common /ABCLibpval/ nprocs, myid, ncelx,ncely, nx, ny, ierrcode
```

common 文における変数は，すべて整数変数とします．また設定する変数の意味は以下のとおりです.

- nprocs：利用するプロセッサ台数
- myid：自分のプロセッサ番号 (0,...,nprocs−1)
- ncelx, ncely： nprocs = ncelx * ncely となる値
- nx, ny:利用する問題サイズをnとするとき，それぞれ, nx = n / ncelx, ny = n / ncely, となる値.

- ierrcode：ライブラリリターン時にエラーコードが入る．値設定不要.

ライブラリコールは，以下の形式でおこないます.

```
call ABCLibDRSAllEigVec(A, n, eig, X, ms, me)
```

各引数の意味は以下のとおりです.

- A ： サイクリック-サイクリック分散された係数行列(0:NdivP-1, 0:NdivP-1)
- n ： 係数行列の次元．整数変数.
- eig ： ブロック分散された，計算された固有値 (1:NdivP)
- X ： 列方向ブロック分散された，計算された固有ベクトル(1:n, 1:NdivP-1)
- ms ： 要求する固有ベクトルに関する，対応する固有値の開始番号 (絶対値の大きいほうから数える)
- me ： 要求する固有ベクトルに関する，対応する固有値の終了番号 (絶対値の大きいほうから数える)

4.2 エルミート行列の全固有値，全固有ベクトルの求解

エルミート行列の全固有値，全固有ベクトルを求めるためのサンプルプログラム (tests/ABCLibTestHerAllEigVec.f) は，以下のとおりです.

```
      program  main
      include  'ABCLibDRSSED.h'
      common  /ABCLibpval/ nprocs, myid, ncelx, ncely, nx, ny, ierrcode

c     === real*8 definition
c     ============================================================
c     ===  for tridiagonalzation
c        === The AR is real part, and the AI is imaginary part of
c               the Hermitian Matrix of A
      real*8  AR(1:ABCLibMAXN, 1:ABCLibMAXN)
      real*8  AI(1:ABCLibMAXN, 1:ABCLibMAXN)

c        === The number of eigenvalues of the matrix A is 2 times
c               larger than original one, because of conjugate eigenvectors.
      real*8  eig(1:ABCLibMAXNDIVP*2)
      real*8  X(1:ABCLibMAXN*2, 1:ABCLibMAXNDIVP*2)

c     === for checking answer
```

```
      real*8  dtemp
      real*8  EigErr, VecErr1, VecErr2, SysErr

c     === for measuring time
      real*8    t1, t2, bt, t_all
c     ============================================================

c     === integer definition
c     ============================================================
c     === for parallel control
      integer   ncelx, ncely, nx, ny
      integer   myid, nmyidx, nmyidy
      integer   n, nprocs

c     === for error flag
      integer   ierrcode
c     === for selecting test matrices
      integer   isw_mat
c     ============================================================

c     -=- Program start
c     =-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=

c     === MPI Init.
c     ====================================================
      call MPI_INIT( ierr )
      call MPI_COMM_RANK( MPI_COMM_WORLD, myid, ierr )
      call MPI_COMM_SIZE( MPI_COMM_WORLD, nprocs, ierr )
c     ====================================================

c     === Print title
c     ====================================================
      if (myid .eq. 0) then
        print *, ""
        print *, "=== Parallel Eigensolver Check Program ==="
        print *, "Parallel Eigensolver on MPI"
        print *, "ABCLibHerAllEigVec"
        call ABCLibPrintVer()
        print *, ""
      endif
c     ====================================================
```

```
c     === Setting Test Condition
c     ==================================================
c     === set matrix type
      isw_mat = 1
      call MPI_BCAST(isw_mat,1,MPI_INTEGER,0,MPI_COMM_WORLD,ierr)
      if (isw_mat .ge. 5) then
        if (myid .eq. 0) then
          write(6,*)  "Problem size of the Hermitian matrix >"
          read(5,*) n
        endif
      endif
c     ==================================================

c     === Set Parallel Control Values
c     ==================================================
      if (myid .eq. 0) then
        write(6,*) " ncelx ="
        read(5,*) ncelx
        write(6,*) " ncely ="
        read(5,*) ncely
      endif
      call MPI_BCAST(ncelx, 1, MPI_INTEGER, 0, MPI_COMM_WORLD, ierr)
      call MPI_BCAST(ncely, 1, MPI_INTEGER, 0, MPI_COMM_WORLD, ierr)
c     ==================================================

c     === Generate Test Matrix
c     ==================================================
      call HerTestMatDim(n, isw_mat)
      call HerMakeTestMat(AR, AI, n, isw_mat)
c     ==================================================

c     === Set Parallel Control Values
c     ==================================================
      call MPI_BCAST(n,1,MPI_INTEGER,0,MPI_COMM_WORLD,ierr)
      nx = ceiling(dfloat(2*n)/dfloat(ncelx))
      ny = ceiling(dfloat(2*n)/dfloat(ncely))
c     ==================================================

c     === Initialize ABCLibHerAllEig routine
c     ==================================================
      call ABCLibHerAllEigVec_Init()
c     ==================================================
```

```
c     === Call Hermitian Eigenvalue Computation Routine
c     ================================================
      call MPI_BARRIER(MPI_COMM_WORLD, ierr)
      t1 = MPI_WTIME()

      call ABCLibHerAllEigVec(AR, AI, n, eig, X)

      call MPI_BARRIER(MPI_COMM_WORLD, ierr)
      t2 = MPI_WTIME()

      t_all = t2 - t1
      call MPI_REDUCE(t_all, bt, 1, MPI_DOUBLE_PRECISION, MPI_MAX, 0,
     &                MPI_COMM_WORLD, ierr)
      t_all = bt
c     ================================================

c     === Check Its Error
c     ================================================
      if (ierrcode .eq. 0) then
        print *, myid, "Normal return"
      else
        print *, "myid=",myid,"Abnormal return: error code",ierrcode
      endif

      if (isw_mat .le. 3) then
        if (myid .eq. 0) then
           print *,
     &       "Check of maximal error for eigenvalues...."
        endif
        call HerChkEigErr(eig, n, EigErr, isw_mat)
        if (myid .eq. 0) then
           if (myid .eq. 0) print *, "End of checking."
        endif
      endif

      if (myid .eq. 0) print *,
     &   "Check of error for eigenvectors...."
      call TestVec(X, 2*n, VecErr1, 2*n)
      call TestHerVec(X, eig, n, VecErr2)
      if (myid .eq. 0) print *, "End of checking."
```

```
      if (myid .eq. 0) print *,
     &   "Check of error for the eigensystem...."
      call TestHerEigSys(AR, AI, eig, X, n, isw_mat, SysErr)
      if (myid .eq. 0) print *, "End of checking."
c     ================================================

c     === Output Results
c     ================================================
      if (myid .eq. 0) then
        print *, ""
        print *, "=== Result "
        print *, "============================================== "
        print *, "n = ", n, " ,Number of processors = ", nprocs
        print *, "Test Matrix No:", isw_mat
        print *, ""
        if (isw_mat .le. 3) then
          print *, "Eigenvalue Max Error = ", EigErr
        endif
        print *, "Eigenvector Error of Real ||X^T X - I|| = ", VecErr1
        print *, "Eigenvector Error of Her ||X^H X - I|| = ", VecErr2
        print *, "Eigensystem Error of Hermitian:"
        print *, "  max_i||A x_i - lambda_i x_i|| = ", SysErr
        write(*, 1000) t_all
 1000   format(" Total Calculation Time = ",F10.3,"  [sec]")
        print *, ""
      endif
c     ================================================

c     === Finalize ABCLibHerAllEig routine
c     ================================================
      call ABCLibHerAllEigVec_Finalize()
c     ================================================

c     ===== MPI finazize
c     ================================================
      call MPI_FINALIZE(ierr)
c     ================================================

c     =-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
      stop
      end
```

4.2.1 サンプルプログラムの説明

ヘッダーファイル，および common 文を，同様に付加してください．
ライブラリコールは，以下の形式でおこないます．

```
call ABCLibHerAllEigVec(AR, AI, n, eig, X)
```

各引数の意味は以下のとおりです．

- AR ： 分散されていない，エルミート行列の係数行列の実数部(1:n, 1:n)
- AI ： 分散されていない，エルミート行列の係数行列の虚数部(1:n, 1:n)
- n ： 係数行列の次元．整数変数．
- eig ： ブロック分散された，計算された固有値 (1:2*NdivP)
- X ： 列方向ブロック分散された，計算された固有ベクトル(1:2*n, 1:2*NdivP-1)

4.3 Gram-Schmidt 法による直交化

Gram-Schmidt 法による直交化をするためのサンプルプログラム
(tests/ABCLibTestQRD.f) は，以下のとおりです．

```
      program  main
      include  'ABCLibDRSSED.h'
      common  /ABCLibpval/ nprocs, myid, ncelx, ncely, nx, ny, ierrcode

c     === real*8 definition
c     ============================================================
      real*8  X(1:ABCLibMAXN, 1:ABCLibMAXNDIVP)

c     === for checking answer
      real*8  dtemp
      real*8  VecErr

c     === for measuring time
      real*8    t1, t2, bt, t_all
c     ============================================================

c     === integer definition
c     ============================================================
c     === for parallel control
      integer   ncelx, ncely, nx, ny
      integer   myid, nmyidx, nmyidy
```

```
      integer   n, nprocs

c     === for error flag
      integer   ierrcode
c     === for selecting test matrices
      integer   isw_mat
c     ============================================================

c     -=- Program start
c     =-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=

c     === MPI Init.
c     ====================================================
      call MPI_INIT( ierr )
      call MPI_COMM_RANK( MPI_COMM_WORLD, myid, ierr )
      call MPI_COMM_SIZE( MPI_COMM_WORLD, nprocs, ierr )
c     ====================================================

c     === Print title
c     ====================================================
      if (myid .eq. 0) then
        print *, ""
        print *, "=== Parallel Eigensolver Check Program ==="
        print *, "Parallel QR Decomposition on MPI"
        print *, "ABCLibDRSAllEigVec"
        call ABCLibPrintVer()
        print *, ""
      endif
c     ====================================================

c     === Setting Test Condition
c     ====================================================
      if (myid .eq. 0) then
        write(6,*)  "problem size >"
        read(5,*) n

c       === set matrix type
        isw_mat = 1
      endif
c     ====================================================
```

```
c     == Set Parameters
c     ==================================================
      call MPI_BCAST(n,1,MPI_INTEGER,0,MPI_COMM_WORLD,ierr)
      call MPI_BCAST(isw_mat,1,MPI_INTEGER,0,MPI_COMM_WORLD,ierr)
c     ==================================================

c     === Set Parallel Control Values
c     ==================================================
      if (myid .eq. 0) then
        write(6,*) " ncelx="
        read(5,*) ncelx
        write(6,*) " ncely ="
        read(5,*) ncely
      endif
      call MPI_BCAST(ncelx, 1, MPI_INTEGER, 0, MPI_COMM_WORLD, ierr)
      call MPI_BCAST(ncely, 1, MPI_INTEGER, 0, MPI_COMM_WORLD, ierr)
      call cid2xy2(myid, ncelx, ncely, nmyidx, nmyidy)
      nx = ceiling(dfloat(n)/dfloat(ncelx))
      ny = ceiling(dfloat(n)/dfloat(ncely))
c     ==================================================

c     === Generate Test Matrix
c     ==================================================
      call MakeMatSeqCB(X, n, isw_mat)
c     ==================================================

c     === Initialize ABCLib_QRD routine
c     ==================================================
      call ABCLibQRD_Init()
c     ==================================================

c     === Call Tridiagonalication Routine
c     ==================================================
      call MPI_BARRIER(MPI_COMM_WORLD, ierr)
      t1 = MPI_WTIME()

      call ABCLib_QRD(X, n)

      call MPI_BARRIER(MPI_COMM_WORLD, ierr)
      t2 = MPI_WTIME()

      t_all = t2 - t1
```

```
      call MPI_REDUCE(t_all, bt, 1, MPI_DOUBLE_PRECISION, MPI_MAX, 0,
     &                MPI_COMM_WORLD, ierr)
      t_all = bt
c     ====================================================

c     === Orthogonalized Vectors & Check Its Error
c     ====================================================
      if (ierrcode .eq. 0) then
        if (myid .eq. 0) print *, "Normal return"
      else
        print *, "myid=",myid,"Abnormal return: error code",ierrcode
      endif

        if (myid .eq. 0) print *,
     &     "Check of error for vectors...."
        call TestVec(X, n, VecErr, n)
        if (myid .eq. 0) print *, "End of checking."
c     ====================================================

c     === Output Results
c     ====================================================
      if (myid .eq. 0) then
        print *, ""
        print *, "=== Result "
        print *, "================================================ "
        print *, "n = ", n, " ,Number of processors = ", nprocs
        print *, "Test Matrix:"
        if (isw_mat .eq. 1) print *, "Random"
        print *, ""
        print *, "Orthogonal Error of ||X^T X - I|| = ", VecErr
        write(*, 1000) t_all
 1000   format(" Total Calculation Time = ",F10.3,"  [sec]")
        print *, ""
      endif
c     ====================================================

c     === Finalize ABCLib_QRD routine
c     ====================================================
      call ABCLibQRD_Finalize()
c     ====================================================

c     ===== MPI finazize
```

```
c     ====================================================
      call MPI_FINALIZE(ierr)
c     ====================================================

c     =-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
      stop
      end
```

4.3.1 サンプルプログラムの説明

ヘッダーファイル，および common 文を，同様に付加してください．
ライブラリコールは，以下の形式でおこないます．

```
call ABCLib_QRD(X, n)
```

各引数の意味は以下のとおりです．

- X：列方向ブロック分散された，直交化すべきベクトルからなる行列(1:n，1:NdivP-1)．なお，直交化済のベクトルもこの行列に上書きされる．
- n ： 係数行列の次元．整数変数．

5 実行例

5.1 密対称実数行列の固有値・固有ベクトル計算に関するライブラリの実行

以下に，密対称実数行列の固有値・固有ベクトル計算を行うテスト用プログラム ABCLibTestDRSAllEigVec.f の実行例をのせます．
なおこの例での指定ファイル .ABCLibDRSAllEigVec (5.4.1 節で説明) は以下の通りです．

```
-ort CGS -autotune no -starttunesize 128 -maxtunesize 512
-tunestride100 128 -tunestride1000 1000 -print yes
```

以下に 4PE を用いて，100 次元の Frank 行列の固有値と固有ベクトルを，固有値の絶対値が大きいほうから数えて 10 番目から 60 番目までの固有値 51 個と，それに対応する固有ベクトル 51 個を求めた結果をのせます．

```
$ mpirun -np 4 -machinefile nodeinfo ABCLibTestDRSAllEigVec

=== Parallel Eigensolver Check Program ===
 Parallel Eigensolver on MPI
 ABCLibDRSAllEigVec
```

```
  ----------------------------------------
  ABCLib ver. release Alpha
  Composed by T. Katagiri, December  2002
  ----------------------------------------
  Problem size >
100
 Start number of eigenvectors >
10
 End number of eigenvectors >
60
  ncelx=
2
  ncely =
2
  !!!! Warning !!!!
  There is no 'autotuneTRD.dat' file.
  I will set an AD-HOC parameters for TriRed.
  I strongly recommend that you do autotuning.
  !!!! Warning !!!!
  There is no 'autotuneHIT.dat' file.
  I will set an AD-HOC parameters for HIT.
  I strongly recommend that you do autotuning.
 Auto-tuned :(Tri)          100            0            8            6
 Auto-tuned (HIT):          100            1            1
 Tridiagonalization Time =      0.298  [sec]
 Re-distribution Time =      0.002  [sec]
 Calculating Eigenvalues Time =      0.003  [sec]
 ------ All Eigenvalues Cal. Time =      0.303  [sec] ------
 Gathering Eigenvalues Time =      0.038  [sec]
 SubMatrix Indexes :             1          100
 Number of Groups :             1
 Max clustered eigenvalues :            51
 Constant for convergence :    4.9652297362006328E-012
 Distance for orthogonalization  :     6.391867118793334
 fact :    1.0000000000000000E-003
 Method for ort. : Peters-Wilkinson
 Inverse Iteration : CGS
 Calculating Eigenvectors Time =      0.629  [sec]
 Householder Inverse Iteration Time =      0.084  [sec]
 myid=            0 Abnormal return: error code         4000
 Check of maximal error for eigenvalues....
 End of checking.
```

```
 Check of error for eigenvectors....
 End of checking.
 Check of error for the eigensystem....
 End of checking.
  === Result
 ================================================
 n =           100  ,Number of processors =             4
 Number of eigenvalues and eigenvectors =            51
 Start Eigenvalue No.           10 / End Eigenvalue No.           60
 Test Matrix:
 Frank
 Eigenvalue Max Error =    5.1249242025868220E-014
 Eigenvector Error of ||X^T X - I|| =    7.7850102906234229E-014
 Eigensystem Error max_i||A x_i - lambda_i x_i|| =    1.6640934173348136E-011
 Total Calculation Time =      1.059  [sec]
```

5.2 エルミート行列の固有値・固有ベクトル計算に関するライブラリの実行

以下に，エルミート行列の全固有値・全固有ベクトル計算を行うテスト用プログラム ABCLibTestHerAllEigVec.f の実行例をのせます.

なおこの例での指定ファイル .ABCLibHerAllEigVec (5.4.1 節で説明) は以下の通りです.

```
-ort MGS -autotune no -starttunesize 128 -maxtunesize 512
-tunestride100 128 -tunestride1000 1000 -print yes
```

以下に 4PE を用いた結果をのせます.

```
$ mpirun -np 4 -machinefile nodeinfo ABCLibTestHerAllEigVec

 === Parallel Eigensolver Check Program ===
 Parallel Eigensolver on MPI
 ABCLibHerAllEigVec
  ------------------------------------------
  ABCLib ver. release Alpha
  Composed by T. Katagiri, December  2002
  ------------------------------------------

  ncelx =
2
  ncely =
2
```

```
 !!!! Warning !!!!
 There is no 'autotuneTRD.dat' file.
 I will set an AD-HOC parameters for TriRed.
 I strongly recommend that you do autotuning.
 !!!! Warning !!!!
 There is no 'autotuneHIT.dat' file.
 I will set an AD-HOC parameters for HIT.
 I strongly recommend that you do autotuning.
Auto-tuned :(Tri)            8            0            8            6
Auto-tuned (HIT):            8            1            1
Tridiagonalization Time =      0.965  [sec]
Re-distribution Time =      0.007  [sec]
Calculating Eigenvalues Time =      0.002  [sec]
------ All Eigenvalues Cal. Time =      0.974  [sec] ------
Gathering Eigenvalues Time =      0.038  [sec]
SubMatrix Indexes :             1            3
Number of Groups :             3
Max clustered eigenvalues :             0
Constant for convergence :    1.2414855315468873E-014
Distance for orthogonalization  :    2.7931648995851659E-002
fact :    1.0000000000000000E-003
Method for ort. : Peters-Wilkinson
Inverse Iteration : MGS
Calculating Eigenvectors Time =      0.000  [sec]
Householder Inverse Iteration Time =      0.003  [sec]
myid=            0 Abnormal return: error code         4000
Check of maximal error for eigenvalues....
End of checking.
Check of error for eigenvectors....
End of checking.
Check of error for the eigensystem....
End of checking.

=== Result
================================================
n =             4  ,Number of processors =             4
Test Matrix No:            1

Eigenvalue Max Error =    1.4924678768188513E-015
Eigenvector Error of Real ||X^T X - I|| =    2.9447165805096462E-011
Eigenvector Error of Her ||X^H X - I|| =    3.7867080775839170E-011
Eigensystem Error of Hermitian:
```

```
   max_i||A x_i - lambda_i x_i|| =    4.9365182464366839E-011
 Total Calculation Time =      1.062  [sec]
```

5.3 Gram-Schmidt 法による直交化に関するライブラリの実行

以下に，Gram-Schmidt 法による直交化を行うテスト用プログラム ABCLibTestQRD.f の実行例をのせます.

なおこの例での指定ファイル.ABCLibQRD (5.4.1 節で説明) は以下の通りです.

```
-ort MGS -autotune no -starttunesize 128 -maxtunesize 512
-tunestride100 128 -tunestride1000 1000 -print yes
```

以下に 4PE を用いて，100 次元の乱数ベクトル 100 個を直交化させる実行例をのせます.

```
$ mpirun -np 4 -machinefile nodeinfo ABCLibTestQRD

 === Parallel Eigensolver Check Program ===
 Parallel QR Decomposition on MPI
 ABCLibDRSAllEigVec
  ------------------------------------------
  ABCLib ver. release Alpha
  Composed by T. Katagiri, December  2002
  ------------------------------------------

 problem size >
100
  ncelx=
2
  ncely =
2
  !!!! Warning !!!!
  There is no 'autotuneMGSAO.dat' file.
  I will set AD-HOC parameters for TriRed.
  I strongly recommend that you do autotuning.
 Auto-tuned :            4            4            8            4            8
 Tridiagonalization Time =      0.798  [sec]
 ----- Orthogonalization Time =      0.798  [sec] -----
 Normal return
 Check of error for vectors....
 End of checking.

 === Result
 ================================================
```

```
 n =           100   ,Number of processors =             4
 Test Matrix:
 Random

 Orthogonal Error of ||X^T X - I|| =    5.8215990320764321E-013
 Total Calculation Time =      0.862  [sec]
```

5.4 自動チューニングの実行

5.4.1 自動チューニングの指定方法と指定ファイルの内容

ABCLibDRSSED における自動チューニングは，各ライブラリルーチン毎に決められた名前をもつファイル (指定ファイル) で指示します.

以下に，各ライブラリルーチンと対応する指定ファイル名をのせます.

- 実数対称密行列用三重対角化ルーチン [一般版] (ABCLibTriRed)：
  指定ファイル .ABCLibTriRed
- 実数対称密行列用三重対角化ルーチン [特化版] (ABCLibTriRedR)：
  指定ファイル .ABCLibTriRedR
- 実数対称密行列用固有値求解ルーチン (ABCLibDRSAllEig)：
  指定ファイル .ABCLibDRSSEDAllEig
- 実数対称密行列用固有値および固有ベクトル求解ルーチン (ABCLibDRSAllEigVec)：
  指定ファイル .ABCLibDRSSEDAllEigVec
- エルミート行列用全固有値求解ルーチン (ABCLibHerAllEig)：
  指定ファイル .ABCLibHerAllEig
- エルミート行列用全固有値および全固有ベクトル求解ルーチン (ABCLibHerAllEigVec)：
  指定ファイル .ABCLibHerAllEigVec
- Gram-Schmidt 直交化ルーチン (ABCLibQRD)：
  指定ファイル .ABCLibQRD

また指定ファイルに記述できる内容は，以下の通りです．なお指定ファイルには，以下の記述を含め，最低 80 文字以上の空白を入れてください.

- -autotune
  自動チューニングするかどうかの指定です．“yes”, “no” で指定し，実装されている自動チューニング全てを行います.
  また，全て自動チューニングしたい場合は “1”，Householder 三重対角化部のみ自動チューニングしたい場合は “2”，Householder 逆変換による固有ベクトル計算部のみ自動チューニングしたい場合は “3”，および，Gram-Schmidt 直交化ルーチンのみ自動チューニングしたい場合は “4”，という指定もできます.

- `-starttunesize`
  自動チューニングする際の，開始する行列次元数を指定します．

- `-maxtunesize`
  自動チューニングする際の，終了する行列次元数を指定します．

- `-tunestride100`
  インストール時自動チューニングする際の，1,000 次元に満たない次元についての増加サイズを指定します．

- `-tunestride1000`
  インストール時自動チューニングする際の，1,000 次元を越える次元についての増加サイズを指定します．

- `-ort`
  固有ベクトル計算の際の，直交化方式を指定します．ここで指定される直交化方式は，固有ベクトルの精度および実行時間に影響を及ぼします．
  リリース版αで提供されている直交化方式の種類は，以下の通りです．

  - `MGS` ：修正 Gram-Schmidt 法による直交化です．最も精度が高い方法ですが，並列性がなく，高速処理に向きません．実行時間にかかわらず精度を保証したい場合に，指定を推奨します．
  - `CGS` ：古典 Gram-Schmidt 法による直交化です．精度が低い方法ですが，並列性があり，高速処理に向いています．
  - `HCGS` ：古典 Gram-Schmidt 法と修正 Gram-Schmidt 法を混合した方法です．古典 Gram-Schmidt 法を用いているため精度が低いのですが，単純な CGS に比べて精度改善される場合があります．並列性があり，高速処理に向いています．
  - `IRCGS` ：古典 Gram-Schmidt 法を 2 回行う直交化方式です．精度が低い方法ですが，単純な CGS に比べ精度改善される場合があります．並列性があり，高速処理に向いています．
  - `SCGS` ：古典 Gram-Schmidt 法による直交化です．精度が低い方法であり，理論上 CGS 法よりも精度が悪くなります．この方式は比較検討などの数値実験の用途以外，普通は指定してはいけません．
  - `NoOrt` ：全く直交化をしない方法です．精度破綻する場合がありますが，並列性が極めて高いので高速実行に向いています．精度はともかく高速に計算したい場合のみ，指定が推奨されます．

- `-print`
  内部実行時間などの詳細情報を表示するかどうかを指定します．表示する場合 “yes”，しない場合 “no” で指定します．

以下に，指定ファイルの記述例をのせます．

例 1：
```
-autotune yes -starttunesize 100 -maxtunesize 8000
-tunestride100 100 -tunestride1000 1000 -ort MGS -print yes
```

例 1 の記述では，行列サイズ 100 次元から 1,000 次元までは 100 刻みで，1,000 次元を越える場合は 1,000 刻みで，8,000 次元までの行列サイズをサンプル点にとり，自動チューニングします．

例 2：`-autotune no -starttunesize 100 -maxtunesize 8000 -tunestride100 100 -tunestride1000 1000 -ort MGS -print yes`

例 2 の記述では，直交化方式に MGS，ライブラリの内部実行時間を表示しつつ，ライブラリを実行することを指定します．自動チューニングはしないことを指定します．

5.4.2 自動生成されるファイルの説明

ABCLibDRSSED リリース版 α では，Householder 三重対角化部，Householder 逆変換による固有ベクトル計算部，Gram-Schmidt 直交化処理，の 3 部分に自動チューニング機能が付加されています．

自動チューニングをすると，最適化されたパラメタに関する情報を記述したファイルが自動生成されます．これをパラメタ情報ファイルと呼びます．パラメタ情報ファイルは，自動チューニング対象のライブラリルーチンごとに生成されます．

これらパラメタ情報ファイルは，以下に示す通りです．

Householder 三重対角化部について：

- `autotuneTRD.dat`
  最適な行列更新のアンローリング段数，行列ベクトル積のアンローリング段数，通信方式，に関するパラメタ保存用ファイル

- `TrdUpdLSM.dat`
  最適な行列更新のアンローリング段数に関する，最小二乗法による係数情報

- `TrdMatVecLSM.dat`
  最適な行列ベクトル積のアンローリング段数に関する，最小二乗法による係数情報

- `TrdCommLSM.dat`
  最適な通信方式に関する，最小二乗法による係数情報

- `autotuneTRDanal.dat`
  自動チューニングに関する，解析結果保存用ファイル

Householder 逆変換による固有ベクトル計算部について：

- `autotuneHIT.dat`
  最適な行列更新のアンローリング段数，通信方式，に関するパラメタ保存用ファイル

- `HITKerLSM.dat`
  最適な行列更新のアンローリング段数に関する，最小二乗法による係数情報

- `HITCommLSM.dat`
  最適な通信方式に関する，最小二乗法による係数情報

- autotuneHITanal.dat
  自動チューニングに関する，解析結果保存用ファイル

Gram-Schmidt 直交化処理について：

- autotuneMGSAO.dat
  最適なブロック幅，アンローリング段数，に関するパラメタ保存用ファイル

- MGSAOblLSM.dat
  最適なブロック幅に関する，最小二乗法による係数情報

- autotuneMGSAOanal.dat
  自動チューニングに関する，解析結果保存用ファイル

なお最小二乗法に関する詳細な指定の変更は，include/ABCLibLSM.h で指定可能です.

5.4.3 インストール時自動チューニングの実行

インストール時自動チューニングとは，ライブラリインストール時に行う最適化処理のことです.

前節で説明した，各ライブラリルーチンに対応する指定ファイルを用意し処理を指定したうえで，各ライブラリルーチンを呼び出すことでインストール時自動チューニングを行います.

なおインストール時自動チューニングをすることで，前節で説明したパラメタ情報ファイルが自動生成されます．このファイルを参照することで，最適なパラメタが自動設定されます．したがってインストール時自動チューニングは基本的に，ライブラリのインストール後 1 回おこなえば十分です¶ .

[**実行例**]

以下のように，.ABCLibDRSAllEigVec を設定した場合のインストール時自動チューニングの実行例をのせます.

```
-ort MGS -autotune yes -starttunesize 128 -maxtunesize 512
-tunestride100 128 -tunestride1000 1000 -print yes
```

上記のファイルを実行ディレクトリにおいたあと，以下のコマンドを実行します.

```
$  mpirun -np 4 -machinefile nodeinfo ABCLibTestDRSAllEigVec
```

上記のコマンドを実行後，問題サイズなど入力を求めてくるので，以下のように入力します．なお問題サイズ，求める固有値の開始番号，および終了番号については，.ABCLibDRSAllEigVec ファイルの値が利用されるので，任意の数値で結構です.

¶ 利用するプロセッサ数を変えた場合，利用する通信網などのハードウエア構成を変えた場合，などもインストール時自動チューニングを再度実行する必要があります.

```
 Problem size >
100
 Start number of eigenvectors >
2
 End number of eigenvectors >
2
  ncelx=
2
  ncely =
2
```

入力後，以下のようなログが出力され，自動チューニングモードに入ります.

```
  !!!! Warning !!!!
  There is no 'autotuneTRD.dat' file.
  I will set AD-HOC parameters for TriRed.
  I strongly recommend that you do autotuning.
  !!!! Warning !!!!
  There is no 'autotuneHIT.dat' file.
  I will set AD-HOC parameters for HIT.
  I strongly recommend that you do autotuning.
  ===== AUTO Tuning Mode ======
  ------------------------------------------
  ABCLib ver. release Alpha
  Composed by T. Katagiri, December  2002
  ------------------------------------------

 === Auto-tuning for Tridiagonalization ===
  ------------------------------------------
  ABCLib ver. release Alpha
  Composed by T. Katagiri, December  2002
  ------------------------------------------

 Now start tuning...
            1            1 Initial swichies:            0            8
            6
 n=          128 Time=   0.5255729999999992      iupdate_sw=            1
 n=          128 Time=   0.2111440000000004      iupdate_sw=            2
 n=          128 Time=   0.1672940000000001      iupdate_sw=            3
 n=          128 Time=   0.1705660000000000      iupdate_sw=            4
 n=          128 Time=   0.1708849999999993      iupdate_sw=            5
 n=          128 Time=   0.1729319999999990      iupdate_sw=            6
 n=          128 Time=   0.1736940000000011      iupdate_sw=            7
```

```
 n=          128 Time=   0.1667370000000008      iupdate_sw=           8
 n=          128 Time=   0.1737360000000008      iupdate_sw=           9
 n=          128 Time=   0.1685940000000001      iupdate_sw=          10
 n=          128 Time=   0.1730620000000007      iupdate_sw=          11
 n=          128 Time=   0.1696140000000010      iupdate_sw=          12
 n=          128 Time=   0.1680330000000017      iupdate_sw=          13
 n=          128 Time=   0.1737040000000021      iupdate_sw=          14
 n=          128 Time=   0.1699410000000019      iupdate_sw=          15
 n=          128 Time=   0.1740300000000010      iupdate_sw=          16
....
```

自動チューニングが全て終了後，以下のようなファイルが自動生成されています.

```
$ ls
ABCLibTestDRSAllEigVec      autotuneTRDanal.dat       autotuneHITanal.dat
autotuneTRDanal_tmp.dat     MGSAOblLSM.dat            autotuneHITanal_tmp.dat
autotuneTRD.dat             nodeinfo                  autotuneHIT.dat
TrdCommLSM.dat              autotuneMGSAOanal.dat     TrdMatVecLSM.dat
autotuneMGSAOanal_tmp.dat   HITCommLSM.dat            TrdUpdLSM.dat
autotuneMGSAO.dat           HITKerLSM.dat
```

ここで，三重対角化ルーチンの自動チューニング結果 (最適パラメタ) を保存するパラメタ情報ファイルautotuneTRD.dat の中身は，以下のとおりです.

```
  128     1     8     8
  256     0     8     3
  384     0     7     4
  512     0     4     3
   -1     0     8     6
   -1     0     8     6
   -1     0     8     6
   -1     0     8     6
   -1     0     8     6
   -1     0     8     6
   -1     0     8     6
   -1     0     8     6
   -1     0     8     6
   -1     0     8     6
   -1     0     8     6
   -1     0     8     6
   -1     0     8     6
   -1     0     8     6
   -1     0     8     6
   -1     0     8     6
   -1     0     8     6
```

また，Gram-Schmidt 直交化ルーチンの自動チューニングログを解析した結果を保存するファイルautotuneMGSAOanal.dat の中身は，以下のとおりです.

```
=== ABCLib_QRD Tuning Log Analysis Result
 ========================================
 Number of PE:    4/Tuning Time:              104.148 [Sec.]
 ========================================
 ProblemSize  |  Const [Sec.] (Parameter)  |  Worst [Sec.] (Parameter) |
                 Best [Sec.] (Parameter)   |  C/B  |  W/B
    128 |   0.0250(  4,  4,  8,  4,  8) |  0.0798(  1,  4,  8,  4,  8) |
           0.0245(  5,  1,  3,  4,  8) |      1.022 |      3.255
    256 |   0.0940(  4,  4,  8,  4,  8) |  0.0988(  1,  4,  8,  4,  8) |
           0.0937(  5,  1,  1,  3,  2) |      1.003 |      1.055
    384 |   0.2083(  4,  4,  8,  4,  8) |  0.2173(  1,  4,  8,  4,  8) |
           0.2042(  5,  2,  5,  4,  5) |      1.020 |      1.064
    512 |   0.3729(  4,  4,  8,  4,  8) |  0.4359(  1,  4,  8,  4,  8) |
           0.3664(  6,  2,  4,  3,  6) |      1.018 |      1.190
```

5.4.4 実行起動前自動チューニングの実行

実行起動前自動チューニングとは，問題サイズや係数行列の情報が固定された場合に行う，インストール時自動チューニングよりも精度の高い最適化処理のことです.

リリース版αでは，実行起動前自動チューニング用のライブラリルーチンは存在しません．したがって，前節で説明した自動チューニング時に生成されたパラメタ情報ファイルを手動で全て消去した上で，問題サイズを固定するような自動チューニング指定を行うことでこの機能を実現します.

以下に実行起動前自動チューニングの例をのせます.

例 3：`-autotune yes -starttunesize 1234 -maxtunesize 1234 -tunestride100 100 -tunestride1000 1000 -ort MGS -print yes`

例 3 では，-starttunesize と -maxtunesize が 1234 次元で同一なので，1234 次元のみパラメタを最適化します．この自動チューニングの後，実行時に 1234 次元を指定することで，推定でない最適パラメタの設定がなされます.

このことにより，インストール時自動チューニングを用いて最適化した場合よりも高速なライブラリ実行が期待できます.

6 今後の予定

ABCLibDRSSED 開発プロジェクトの今後の予定について，簡単に説明します.

- 実行起動前自動チューニング専用ルーチンの搭載：
  本リリース版αでは，概説のとおり実行起動前自動チューニングは手動となっております．専用ルーチンの開発，およびインストール時自動チューニングの最適パラ

メタ管理と統合，などを行い使いやすさの向上を目指します．また，固有ベクトル計算の際の直交化方式の選択に関する実行起動前自動チューニングルーチンも搭載予定です．

- ブロック化された三重対角化ルーチンの追加：
  リリース版αでは，ブロック化アルゴリズムを用いた三重対角化ルーチンは搭載されていません．PC クラスタなどの階層メモリを有する計算機では，ブロックアルゴリズムによる計算効率の改善を狙うしかありません．今後ブロックアルゴリズムを実装し，さらに自動チューニング項目に追加する予定です．

- 自動チューニング手法の改良：
  本リリース版αでは，評価関数として線形多項式を採用し，最小二乗法にてパラメタの最適化を行っております．このことから，パラメタ推定精度が十分でない場合が生じます．今後さらに精度がよい評価関数を導入することで，パラメタ推定精度の向上を目指します．

- 疎行列反復解法との融合：
  本リリース版αでは，密行列に対する直接解法を相似変換として採用し，また固有ベクトル計算には反復解法を採用しています．行列が疎行列の場合，全ての処理を反復解法にするほうが効率が良いことが知られています．また行列が帯行列の場合，帯行列用の相似変換を採用するほうが効率が良いことが知られています．このことから，入力行列の非零要素の形状を自動的に検知し，適切な解法を自動選択する自動チューニング機能 (実行起動前，および実行時) の研究開発を行い，さらなる高速化／ユーザビリティの改善，を狙います．

- 既存ライブラリとの融和：
  今後，BLAS(Basic Linear Algebra Subprograms) や ARPACK といったパッケージと同様のインターフェースを用意する，および`ABCLib` 内で自動チューニングする項目として採用することで，さらなるユーザビリティや性能の向上を目指します．

おわりに

本マニュアルでは，`ABCLibDRSSED` リリース版αの利用方法について説明しました．

`ABCLib` プロジェクトの今後の情報を入手されたい方は，`http://www.abc-lib.org/`をご参照ください．

参考文献

[1] 片桐孝洋: ABC-LIB：自動ブロック化・通信最適化ライブラリの開発, JSPP'2002 論文集, つくば市，茨城県, pp. 155–156 (2001).

[2] 片桐孝洋, 吉瀬謙二, 本多弘樹, 弓場敏嗣: FIBER：インストール時，実行起動前，実行時の最適化階層を有する自動チューニングソフトウエア構成方式 –固有値計算ルーチンへの適用と評価–, *DRAFT* (2002 年 11 月). 未発表文献.

[3] 片桐孝洋: プログラム、記録媒体およびコンピュータ, 日本国特許出願, 特願 2003-022792 (平成 15 年 1 月 30 日).

[4] Katagiri, T., Kise, K., Honda, H. and Yuba, T.: FIBER: A Framework of Installation, Before Execution-invocation, and Run-time Optimization Layers for Auto-tuning Software, 電気通信大学情報システム学研究科技術報告, Vol. UEC-IS-2003-3 (2003 年 5 月 13 日).

[5] 片桐孝洋: ABCLibScript 利用の手引, *DRAFT* (2002 年 11 月). 未発表文献.

[6] 片桐孝洋: 計算装置、計算方法、プログラムおよび記録媒体, 日本国特許出願, 特願 2003-092592 (平成 15 年 3 月 28 日).