複素対称 (密) 行列対角化ルーチン並列化

2016 年 2 月 29 日
理化学研究所、仁科加速器研究センター
船木 靖郎 (funaki@riken.jp)

• 依頼内容

$N \times N$ 正定値実対称行列 B 及び $N \times N$ 複素対称行列 $A({}^tA = A)$ に対し、$Ax = \lambda Bx$、あるいは $Ax = \lambda x$ の形の固有値問題を解く MPI 版のサブルーチンを開発すること。

• 支援、開発内容

1. 複素対称 (密) 行列に対し、$Ax = \lambda x$ の固有値問題を解き、全ての固有値、固有ベクトルを求める並列版サブルーチンの開発。
   対角化のドライバルーチンは
   PZDIAG(MAJOR, IP, N, A, NP, NB, A2, V2, INFO)
   本ルーチンの引数仕様は Table 1 を参照。

Table 1: PZDIAG 引数仕様

引数	型	種別	引数の説明
MAJOR	Character*1	INPUT	'R' か'C' を選択。 係数行列をブロック分割したプロセスマップの方向。 'R'('C') 行 (列) 方向に連続した MPI プロセス番号を割り当てる。 **部分行列の分散方法は ScaLAPACK 準拠。**
IP	Integer	INPUT	各プロセスのランク。
N	Integer	INPUT	全体行列 A の次元数。$N \geq 2$
A	Complex*16	INPUT	$A(N(N+1)/2)$: 行列 A の要素を 1 次元配列で指定。 $A(I,J) :\Rightarrow A(I(I-1)/2+J)$ $(I \geq J)$, $A(J(J-1)/2+I)$ $(J \geq I)$。
NP	Integer	INPUT	並列化数。 **2 の偶数べき乗に設定すること** $(1, 4, 16, 64, 256, \cdots)$**。**
NB	Integer	INPUT	係数行列をブロック分割した小行列の次元数。 **8 以上の偶数に設定すること**。
A2	Complex*16	OUTPUT	$A2(N)$: 1 次元配列。A の固有値。
V2	Complex*16	OUTPUT	$V2(N,N)$: 2 次元配列。A の固有ベクトル。
INFO	Integer	OUTPUT	リターンコード。

Table 2: リターンコード (INFO)

0	正常終了
-2	$N \geq 2$ が設定されていない
-66	NB が 8 以上の整数でない
-13	MAJOR='R' または MAJOR='C' が設定されていない
100	プロセスマップが正方かつ 1 辺が 2 のべき乗でない

PZDIAG における計算手順 (詳細はパッケージ内 Document/に置かれたマニュアル等参照)

(1). 複素対称 ${}^tA = A$(非エルミート) 行列 A を各プロセスに分配 (MATDIV)。
(2). A をハウスホルダ変換により 3 重対角化する (PZSYTRD)。
(3). ハウスホルダ逆変換により初期固有ベクトルを生成 (PZSYGHR)。
(4). (3). で計算した初期値をランク 0 のプロセスに集める (MARGEV)。
(5). ランク 0 のプロセスで QR 法により固有値、固有ベクトルを逐次計算 (QR)。

PZDIAG を使用したテストプラグラム例はパッケージ (sym_complx_diag.zip) 内の test.f。テストプラグラム中で呼び出している MATGEN_HAMIL は Input の複素対称行列 A を生成するサンプルプログラム。RESID は固有ベクトルの規格直交性、及び残差二乗和 $\sum_i \|Ax^{(i)} - \lambda^{(i)}x^{(i)}\|/|\lambda^{(i)}|$ をチェックするプログラム。

PZDIAG が使用するソースプログラムは SRC/に置かれている。PZDIAG を使用するためのライブラリ (libPZDIAG.a) を作成する makefile の例は make_lib に書かれている。またパッケージ内の makefile には Fortran コンパイラに Intel コンパイラを利用する mpif90 を指定している。

2. 複素対称 (密) 行列に対し、$Ax = \lambda Bx$ の一般化固有値問題を解き、全ての固有値、固有ベクトルを求める並列版サブルーチンの開発。

   対角化のドライバルーチンは
   PZDIAG_GEN(MAJOR, IP, N, A, B, NP, NB, A2, V2, INCX, INFO)
   本ルーチンの引数仕様は Table 3 を参照。**本ルーチン内で、LAPACK、ScaLAPACK のルーチンを呼び出しているため、それらを使用できる環境が必須。**

Table 3: PZDIAG_GEN 引数仕様

引数	型	種別	引数の説明
MAJOR	Character*1	INPUT	'R' か'C' を選択。 係数行列をブロック分割したプロセスマップの方向。 'R'('C') 行 (列) 方向に連続した MPI プロセス番号を割り当てる。 **部分行列の分散方法は ScaLAPACK 準拠。**
IP	Integer	INPUT	各プロセスのランク。
N	Integer	INPUT	全体行列 A の次元数。$N \geq 2$
A	Complex*16	INPUT	$A(N(N+1)/2)$: 行列 A の要素を 1 次元配列で指定。 $A(I,J) :\Rightarrow A(I(I-1)/2+J)$ $(I \geq J)$, $A(J(J-1)/2+I)$ $(J \geq I)$。
B	Real*8	INPUT	$B(N(N+1)/2)$: 行列 B の要素を 1 次元配列で指定。 $B(I,J) :\Rightarrow A(I(I-1)/2+J)$ $(I \geq J)$, $B(J(J-1)/2+I)$ $(J \geq I)$。
NP	Integer	INPUT	並列化数。 **2 の偶数べき乗に設定すること** $(1, 4, 16, 64, 256, \cdots)$**。**
NB	Integer	INPUT	係数行列をブロック分割した小行列の次元数。 **8 以上の偶数に設定すること**。
A2	Complex*16	OUTPUT	$A2(N)$: 1 次元配列。A の固有値。
V2	Complex*16	OUTPUT	$V2(N, INCX)$: 2 次元配列。A の固有ベクトル。
INCX	Integer	INPUT	求める固有ベクトルの個数
INFO	Integer	OUTPUT	リターンコード。

Table 4: リターンコード (INFO)

0	正常終了
-2	$N \geq 2$ が設定されていない
-66	NB が 8 以上の整数でない
-13	MAJOR='R' または MAJOR='C' が設定されていない
100	プロセスマップが正方かつ 1 辺が 2 のべき乗でない

PZDIAG_GEN における計算手順 (詳細はパッケージ内 Document/に置かれたマニュアル等参照)

(1). 正定値実対称行列 B、複素対称 ${}^tA = A$(非エルミート) 行列 A を各プロセスに分配 (それぞれ、MATDIV_R、MATDIV)。
(2). B をコレスキー分解 $B = LL^T$(**ScaLAPACK ルーチン**: PDPOTRF)。
(3). 下三角行列 L の逆行列 L^{-1} を求める (**ScaLAPACK ルーチン**: PDTRTRI)。
(4). A を標準化変換 $C = L^{-1}AL^{-T}$(**ScaLAPACK ルーチン**: PZTRMM)。
(注: $Ax = \lambda Bx \Leftrightarrow CL^Tx = \lambda L^Tx$)
(5). C をハウスホルダ変換により 3 重対角化する (PZSYTRD)。
(6). ハウスホルダ逆変換により初期固有ベクトルを生成 (PZSYGHR)。
(7). (6). で計算した固有ベクトルの初期値をランク 0 のプロセスに集める (MARGEV)。
(8). ランク 0 のプロセスで QR 法により固有値及び (C の) 固有ベクトル V を逐次計算 (QR)。
(9). (2). で計算した下三角行列 L をランク 0 のプロセスに集める (MARGEL)。
(10). $x = L^{-T}V$ により A の固有ベクトルを求める (絶対値の小さな固有値に属するものから INCX 個)(**LAPACK ルーチン**: ZTRMV)。

PZDIAG_GEN を使用したテストプラグラム例はパッケージ (sym_complx_diag.zip) 内の test.f。テストプラグラム中で呼び出している MATGEN_NORM、MATGEN_HAMIL はそれぞれ Input の正定値実対称行列 B、複素対称行列 A を生成するサンプルプログラム。RESID_G は固有ベクトルの規格直交性、及び残差二乗和 $\sum_i \|Ax^{(i)} - \lambda^{(i)}Bx^{(i)}\|/|\lambda^{(i)}|$ をチェックするプログラム。

PZDIAG_GEN が使用するソースプログラムは SRC/に置かれている。PZDIAG_GEN を使用するためのライブラリ (libPZDIAG.a) を作成する makefile の例は make_lib に書かれている。またパッケージ内の makefile には Fortran コンパイラに Intel コンパイラを利用する mpif90 を指定している。