Sakurai-Sugiura法による固有値問題ソルバーz-Paresの解説

永井佑紀

平成26年9月5日

目 次

1 はじめに

Sakurai-Sugiura法 (SS法) による固有値問題ソルバーz-Paresが公開されたので、その日本語解説をここに記しておく。随時アップデートする予定。z-Paresは筑波大学の櫻井先生らが開発したFortran90で書かれたソルバーである。SS法は複素平面上の指定した領域の固有値と固有ベクトルを求めることができる手法である。その理論の詳細はノート「Sakurai-Sugiura 法による行列の対角化」にまとめてあるのでそちらを参照すること。

`http://zpares.cs.tsukuba.ac.jp/`

からダウンロードできる。公式のマニュアルはこちらからダウンロードできる（英語）ので、詳細はこちらを参照すること。

この日本語解説は、このまま通りに入力するとSS法を試せるように、というコンセプトで書いている。密行列用と疎行列用のサンプルコードを載せた。疎行列用は、疎行列に慣れていない人のために、行列要素を計算するサブルーチンを用意すれば勝手に変換して計算してくれるようにした。

2　インストール方法：密行列用

2.1　Mac OS 用シングル CPU

Mac OS 10.9.4 で確認。
インストールするには、まず、解凍：

```
tar zxf zpares_0.9.6.tar.gz
```

し、

```
cd zpares_0.9.6
```

移動する。次に、Mac 用の Makefile を Mekefile.inc からコピー

```
cp Makefile.inc/make.inc.gfortran.seq.macosx make.inc
```

する。デフォルトでは、gfortran を用いてコンパイルする形になっている。また、make.inc 内の

```
USE_MPI = 1
```

を

```
USE_MPI = 0
```

に変更し、MPI なしで動くようにする。そして、

```
make
```

でインストールできる。
なお、シングル CPU 版がきちんとコンパイルできたかどうかを調べるためには、examples ディレクトリのサンプルコードを実行すればよく、ちゃんとコンパイルされていれば ex1 と ex4 と ex5 が動くはずである。つまり、

```
./examples/ex1
```

や

```
./examples/ex4
```

や

```
./examples/ex5
```

はちゃんとエラーなしで実行できるはずである。

2.2　Linux 用 MPI 並列

インストールするには、まず、解凍：

```
tar zxf zpares_0.9.6.tar.gz
```

し、

```
cd zpares_0.9.6
```

移動する。次に、MPI 並列用の Makefile を Mekefile.inc からコピー

```
cp Makefile.inc/make.inc.par make.inc
```

する。この make.inc を自分の環境に応じて書き換える。手元の intel コンパイラによる MPI 並列の場合には、

```
FC = mpiifort
USE_MPI = 1
FFLAG = -O2
LFFLAG =
BLAS =
LAPACK = -lmkl_intel_lp64 -lmkl_intel_thread  -lmkl_core -liomp5 -lpthread
USE_MUMPS = 0
MUMPS_DIR =
MUMPS_DEPEND_LIBS =
```

とした。そして、

```
make
```

とする。うまくいけば、examples ディレクトリに ex1,ex2,ex3,ex4,ex5 が作成される。ex3 は MPI による並列計算のスピードアップを見ることができる。たとえば、4 コア以上ある計算機の上で

```
mpirun -np 4 ./examples/ex3
```

を実行すると、4 並列で計算が行われる。この並列化では、台形公式による数値積分を並列化する。

3　固有値問題を解くためのサンプル：密行列用

サンプルコードは examples ディレクトリに入っているので、こちらとマニュアルを見ることで使い方を学習することができる。この日本語解説では、すでに具体的な行列を持っている場合にどうやって z-Pares を使うかについて述べる。特に、既存のプログラムで用いている対角化ソルバの部分を置き換える形で z-Pares を使うやり方について述べる。

なお、z-Pares は一般化固有値問題に対応しているが、この解説では通常の固有値問題

$$\hat{A}\boldsymbol{x}_i = \lambda_i \boldsymbol{x}_i \tag{1}$$

を解くことを考える。ここで、$\hat{A}$ は $n \times n$ の行列であり、λ_i は i 番目の固有値、$\boldsymbol{x}_i$ は固有ベクトルである。$\hat{A}$ がエルミート行列であれば λ_i は実数であるが、一般的な対角化可能な $\boldsymbol{A}$ の場合には λ_i は複素数である。

この固有値問題の固有値は複素平面上に散らばっている。この複素平面上で楕円領域を考え、その楕円内に入っている固有値を取り出すことができるのが SS 法である。今回は、$\hat{A}$ がエルミート行列である場合を考え、実軸上の固有値を取り出すことを考えよう。このとき z-Pares で指定すべき値は、楕円の左端と右端の複素平面上での座標値（left および right) と、楕円のアスペクト比 b/a である [1]。エルミート行列の場合、left が求めたい固有値の下限、right が上限となる。この場合には専用のサブルーチンがあり、emin および emax を下限および上限の値として設定する。もし行列が複素数のエルミート行列であれば、zpares_zdnshegv 、実数の対称行列であれば zpares_ddnssygv というサブルーチンをそれぞれ用いることになる。

$n \times n$ 行列の場合、計算した結果返ってくるのは、

1. num_ev: 指定した楕円領域に入っていた固有値の数
2. eigval: 固有値（eigval(1:num_ev)）
3. X: 固有ベクトルが num_ev （X(n, 1:num_ev)）

である。

必要最小限の引数にしたサンプルコードを以下に貼付けておく。このコードは examples の ex4 を参考に作成した。サブルーチン zpares_zdnsgegv_sub の後半の引数 L,N,M,LMAX はオプショナル属性をつけているため、省略が可能である。したがって、

```
call zpares_zdnsgegv_sub('L',A,LDA,emin, emax, num_ev, eigval, X, res)
```

のように呼べば下限 emin 上限 emax の範囲にある行列 A の固有値を求めることができる。このコード test.f90 では z-Pares と BLAS と LAPACK を使用している。もし、gfortran であれば、lib ディレクトリと include ディレクトリの libzpares.a と zpares.mod をプログラムと同じディレクトリにコピー

```
cp ~/zpares_0.9.6/lib/libzpares.a ./
cp ~/zpares_0.9.6/include/zpares.mod ./
```

して、

```
gfortran -L./ -lzpares -framework veclib zpares_sub.f90
```

[1] 楕円はマニュアル Figure 2.3 を参照。

でコンパイルすることができる (Mac OS X の場合)。ここで-framework veclib は BLAS および LAPACK のリンクである。できあがった a.out を実行して、

```
  1   1.4975200987617023       2.50974214070771017E-010
  2   1.5096689223382542       1.59564096980154718E-010
  3   1.5218370261483687       2.36941946038032140E-011
  4   1.5340239317318938       9.33807607329428633E-011
  5   1.5462291598893680       7.01489018103109137E-011
  6   1.5584522307008808       9.65245898792290779E-011
  7   1.5706926635449228       6.72870310480119470E-011
  8   1.5829499771173030       2.00546542472823678E-010
  9   1.5952236894500618       2.73994219290011585E-011
 10   1.6075133179304280       2.04044367536882424E-011
 11   1.6198183793197947       6.53200507140903308E-011
 12   1.6321383897727137       2.39645487341045526E-011
 13   1.6444728648559370       6.30518102985057946E-011
 14   1.6568213195674495       3.94772937731041811E-011
 15   1.6691832683555428       5.85545293276040200E-011
 16   1.6815582251379158       2.87290980487235328E-011
 17   1.6939457033207772       6.94439586125153191E-011
 18   1.7063452158179866       6.79864611400017485E-011
 19   1.7187562750702041       5.91194673675141694E-011
 20   1.7311783930640579       3.82479260925344869E-011
 21   1.7436110813513424       7.44593910629221279E-011
 22   1.7560538510682167       8.96504490099454536E-011
 23   1.7685062129544280       1.20212290896249142E-010
 24   1.7809676773725522       3.41329150809663434E-011
 25   1.7934377543272466       4.33696094557232307E-011
 26   1.8059159534845188       3.06187265792835281E-011
 27   1.8184017841909983       2.21825504926897398E-011
 28   1.8308947554932424       1.27087553752081072E-011
 29   1.8433943761570273       4.57840993113167359E-011
 30   1.8559001546866769       7.12427723371203167E-011
 31   1.8684115993443791       2.02004940175491985E-011
 32   1.8809282181695246       2.67908136956703266E-011
 33   1.8934495189980536       3.13796040795754373E-011
 34   1.9059750094818060       2.77154381021159270E-011
 35   1.9185041971078762       2.89974418720346326E-011
 36   1.9310365892179906       2.18281816854372989E-011
 37   1.9435716930278675       3.15802946808159679E-012
 38   1.9561090156466028       4.18006172304913521E-011
 39   1.9686480640960438       2.20423512800461216E-011
 40   1.9811883453301753       5.62691915476354559E-011
 41   1.9937293662545141       2.18912563131587436E-011
 42   2.0062706337454852       8.06583175721193126E-012
```

となれば問題なく動いていることになる。入力する行列 A を変更すれば、好きなエルミート行列で固有値固有ベクトルを計算することができる。

なお、Linux の MPI 並列版の場合は、

```
mpiifort zpares_sub\_mpi.f90 -L./ -lzpares -lmkl_intel_lp64 -lmkl_intel_thread  -lmkl_core -liomp5 -lpthread
```

などとすればよい。mpiifort は、計算機によっては mpif90 だったりする場合がある。MPI 並列版では、zpares_sub.f90 に MPI の初期化ルーチンと終了ルーチンを付け加え、

```
prm%high_comm = MPI_COMM_WORLD
prm%low_comm = MPI_COMM_SELF
```

を追加する。

ソースコード 1: zpares_sub.f90

```
1
2 module zpares_sub
```

```
3  contains
4    subroutine zpares_zdnsgegv_sub(UPLO,A,LDA,emin, emax, num_ev, eigval, X, res,L,N,M,LMAX)
5      use zpares
6      implicit none
7      character(1),intent(in)::UPLO
8      integer,intent(in)::LDA
9      complex(8),intent(in)::A(1:LDA,1:LDA)
10     real(8),intent(in)::emin,emax
11     integer,intent(out)::num_ev
12     real(8),allocatable,intent(out)::eigval(:),res(:)
13     complex(8),allocatable::X(:,:)
14     integer,optional::L,N,M,LMAX
15 ,      ,!,local, ,variables
16     type(zpares_prm) :: prm
17     integer::i,j,ncv,info
18     complex(8)::B(1:LDA,1:LDA)
19     B = 0d0
20     do i = 1,LDA
21       B(i,i) = 1d0
22     end do
23
24     call zpares_init(prm)
25
26     if ( present(L) ) then
27       prm%L = L
28     else
29       prm%L = 8
30     end if
31     if ( present(N) ) then
32       prm%N = N
33     else
34       prm%N = 32
35     end if
36     if ( present(M) ) then
37       prm%M = M
38     else
39       prm%M = 16
40     end if
41     if ( present(Lmax) ) then
42       prm%Lmax = Lmax
43     else
44       prm%Lmax = 32
45     end if
46
47     ncv = zpares_get_ncv(prm)
48     allocate(eigval(ncv), X(LDA, ncv), res(ncv))
49
50     call zpares_zdnshegv(prm, UPLO, LDA, A, LDA, B, LDA, emin, emax, num_ev, eigval, X, res, info)
51     if(info .ne.0) then
52       write(*,*) "error␣info␣=␣",info
53       stop
54     end if
55
56     call zpares_finalize(prm)
57
58
59     return
60   end subroutine zpares_zdnsgegv_sub
61 end module zpares_sub
62
63 program main
64   use zpares_sub
65   implicit none
66   complex(8),allocatable::A(:,:)
67   integer::LDA,num_ev
68   real(8)::emin,emax
69   real(8), allocatable :: res(:), eigval(:)
70   complex(8), allocatable :: X(:,:)
71   integer::i
72
73   LDA = 500
74
75   allocate(A(1:LDA,1:LDA))
76   call make_matrix(A,LDA)
```

```
77
78    emin = 1.49d0
79    emax = 2.01d0
80
81    call zpares_zdnsgegv_sub('L',A,LDA,emin, emax, num_ev, eigval, X, res)
82
83    do i = 1, num_ev
84       write(*,*) i, eigval(i), res(i)
85    end do
86
87  end program main
88
89  subroutine make_matrix(A,LDA)
90    implicit none
91    integer,intent(in)::LDA
92    complex(8),intent(out)::A(1:LDA,1:LDA)
93  ,   ,!,local, ,variables
94    integer::i,j
95
96    A = (0d0,0d0)
97    do i = 1, LDA
98       do j = 1, LDA
99          if ( i == j ) then
100            A(i,j) = (2d0,0d0)
101         else if ( abs(i-j) == 1 ) then
102            A(i,j) = (1d0,0d0)
103         end if
104      end do
105   end do
106
107   return
108 end subroutine make_matrix
```

4 疎行列の場合

4.1 MUMPS のインストール：逐次版

以下は Linux で intel コンパイラを使っている場合。
まず、MUMPS の入手をする。

```
http://mumps.enseeiht.fr
```

に行き、Download ページの Download request submission に必要事項を入れて Send する。比較的早くメールが返ってくるので、そのメールの指示に従ってソースコードを入手する。
そして、解凍

```
tar zxf MUMPS_4.10.0.tar.gz
```

移動

```
cd MUMPS_4.10.0
```

そして Makefile のコピー

```
cp Make.inc/Makefile.INTEL.SEQ ./Makefile.inc
```

を行う。今回は MUMPS は逐次実行版を入れる。
そして、Makefile.inc を編集し、正しい BLAS の場所を教える。例えば

```
LIBBLAS = -lmkl_intel_lp64 -lmkl_intel_thread -lmkl_core -liomp5 -lpthread
```

とする。
そして make するのだが、ここで z-Pares のインストールで必要なライブラリをすべて作成するため、

```
make alllib
```

とする。これで MUMPS のインストールが完了。

4.2 z-Pares のインストール：逐次版

基本的な手順は密行列版と同じ。違うのは、make.inc ファイルで

```
cp ./Makefile.inc/make.inc.par ./
```

をコピーして、

```
FC = ifort
USE_MPI = 0
FFLAG = -O2
LFFLAG =
BLAS =
LAPACK = -lmkl_intel_lp64 -lmkl_intel_thread  -lmkl_core -liomp5 -lpthread
USE_MUMPS = 1
MUMPS_DIR = /home/nagai/MUMPS_4.10.0
MUMPS_DEPEND_LIBS =
```

のような書き換えを行う。その際、nagai をユーザー名に置き換える等して、MUMPS_4.10.0 の場所を指定する。
そして、

```
make
```

をすればよい。ex6 が疎行列版のサンプルコードである。他のサンプルコードよりもはるかに速いことを確認する。

4.3 実装方法：逐次版

基本的には密行列版と同じ。異なるのは、行列の指定方法だけ。疎行列の指定には CSR (Compressed Sparse Row) を用いている。呼び出しは

```
 zpares_dmpssygv(prm, mat_size, rowA, colA, valA, rowB, colB, valB &
       , emin, emax, num_ev, eigval, X, res, info)
```

である。row,col,val の三つの配列は、CSR 形式の場合に非常によく用いられる形式で、intel の MKL のマニュアルの後半のあたりに例とともに説明が詳しく記述されているので、そちらを参照すること。rowA,colA,valA は解きたい行列 A の情報、rowB,colB,valB を単位行列のものにしておけば、線形固有値問題を解くことができる。サンプルコード zpares_sub.f90 の行列入力を疎行列用に書き換えればそのまま動く。この解説末尾にサンプルコードを載せた。

4.4 MUMPS のインストール：MPI 版

以下は Linux で intel コンパイラを使っている場合。
まず、MUMPS の入手をする。

```
http://mumps.enseeiht.fr
```

に行き、Download ページの Download request submission に必要事項を入れて Send する。比較的早くメールが返ってくるので、そのメールの指示に従ってソースコードを入手する。
そして、解凍

```
tar zxf MUMPS_4.10.0.tar.gz
```

移動

```
cd MUMPS_4.10.0
```

そして Makefile のコピー

```
cp Make.inc/Makefile.INTEL.PAR ./Makefile.inc
```

する。そして、Makefile.inc の一部を

```
CC = mpicc
FC = mpiifort
FL = mpiifort
AR = ar vr
#RANLIB = ranlib
RANLIB  = echo
#SCALAP  = /local/SCALAPACK/libscalapack.a /local/BLACS/LIB/blacs_MPI-LINUX-0.a
  /local/BLACS/LIB/blacsF77init_MPI-LINUX-0.a  /local/BLACS/LIB/blacs_MPI-LINUX-0.a
SCALAP  = -lmkl_scalapack_lp64 -lmkl_blacs_intelmpi_lp64  -lmkl_intel_lp64
-lmkl_intel_thread  -lmkl_core -liomp5 -lpthread -lmkl_scalapack_lp64 -lmkl_blacs_intelmpi_lp64
#INCPAR = -I/usr/local/include
INCPAR =
# LIBPAR = $(SCALAP)  -L/usr/local/lib/ -llamf77mpi -lmpi -llam
LIBPAR = $(SCALAP)   -lmpi -lutil -ldl -lpthread
#-L/usr/local/lib/ -llammpio -llamf77mpi -lmpi -llam -lutil -ldl -lpthread
#LIBPAR = -lmpi++ -lmpi -ltstdio -ltrillium -largs -lt
INCSEQ = -I$(topdir)/libseq
LIBSEQ  =  -L$(topdir)/libseq -lmpiseq
#LIBBLAS = -L/usr/lib/xmm/ -lf77blas -latlas
LIBBLAS = -lmkl_intel_lp64 -lmkl_intel_thread -lmkl_core -liomp5 -lpthread
#LIBBLAS = -L/local/BLAS -lblas
```

と書き換える。CC と FC と FL を書き換えているが、これは MPI のコンパイラが mpiifort の場合であり、mpif90 を使う環境もある。SCALAP と INCPAR と LIBBLAS は、それぞれ MKL のサブルーチンを使うように変更してある。SCALAP は見やすさの問題から上では改行しているが、実際は改行せずに二行を一行にする。

以上の書き換えを行ったあと、

```
make alllib
```

とすればコンパイルができる。

4.5 z-Pares のインストール：MPI 版

基本的な手順は密行列版と同じ。違うのは、make.inc ファイルで

```
cp ./Makefile.inc/make.inc.par ./
```

をコピーして、

```
FC = mpiifort
USE_MPI = 1
FFLAG = -O2
LFFLAG =
BLAS =
LAPACK =  -lmkl_intel_lp64 -lmkl_intel_thread  -lmkl_core -liomp5 -lpthread
USE_MUMPS = 1
MUMPS_DIR = /home/nagai/MUMPS_MPI/MUMPS_4.10.0
MUMPS_DEPEND_LIBS =
```

のような書き換えを行う。その際、nagai をユーザー名に置き換える等して、MUMPS_4.10.0 の場所を指定する。

そして、

```
make
```

をすればよい。ex6 が疎行列版のサンプルコードである。

4.6 実装方法:MPI 版

lib ディレクトリと include ディレクトリの libzpares.a と libzpares_mumps.a と zpares.mod と zpares_mumps.mod をプログラムと同じディレクトリにコピー

```
cp ~/zpares_0.9.6/lib/libzpares* ./
cp ~/zpares_0.9.6/include/zpares* ./
```

し、scalapack 用ダミー

```
cp ../zpares_0.9.6/examples/blacs_scalapack_dummy.o ./
```

をコンパイルは

```
mpiifort zpares_subCRS_mpi.f90 -L./ -lzpares -lzpares_mumps -L./lib -lcmumps -lzmumps
-lmumps_common -lpord -lmkl_intel_lp64 -lmkl_intel_thread -lmkl_core -liomp5 -lpthread blacs_scalapack_dummy.o
```

などとすればよい。-L./lib は MUMPS の lib の場所を指定する。

5 固有値問題を解くためのサンプル：疎行列用

疎行列用のサンプルコードを作成したので、こちらに載せておく。コードは実対称行列用である。このサンプルコードでは線形方程式の求解が速すぎて MPI 並列する必要がない状態 (MPI 並列しても速度が上がらない状態) になっている。

疎行列の CRS フォーマットに慣れていない人のために、(i, j) 番目の行列要素 v を計算するサブルーチンを用意すれば自動的に CRS フォーマットに変換して計算できるようにした。変換用のサブルーチンは make_crs である。このサブルーチンは引数として sub_ijv という任意のサブルーチンを使うことができる。サンプルコードでは make_matrix というサブルーチンがあるが、これを自分の計算したい行列のサブルーチンに書き換えれば動く。

ソースコード 2: zpares_subCRS_mpi.f90

```
1  module zpares_subCRS
2  contains
3    subroutine zpares_dmpssygv_sub(LDA, rowA, colA, valA &
4          , emin, emax, num_ev, eigval, X, res, L,N,M,LMAX)
5      use zpares
6      use zpares_mumps
7      implicit none
8      include 'mpif.h'
9      integer,intent(in)::LDA
10     real(8),intent(in) :: emin, emax
11     integer,intent(in) :: rowA(:), colA(:)
12     real(8),intent(in) :: valA(:)
13     real(8),allocatable,intent(out)::res(:), eigval(:), X(:,:)
14     integer,allocatable::rowB(:),colB(:)
15     real(8),allocatable::valB(:)
16     integer,intent(out)::num_ev
17
18     integer,optional::L,N,M,LMAX
19 ,!,local, ,variables
20     integer::ierr,i,info,ncv
21     type(zpares_prm) :: prm
22
23
24     allocate(rowB(LDA+1),colB(LDA),valB(LDA))
25
26     do i = 1, LDA
27        rowB(i) = i
28        colB(i) = i
29        valB(i) = 1d0
30     end do
31     rowB(LDA+1) = LDA+1
32
33
34     call zpares_init(prm)
35
36     if ( present(L) ) then
37        prm%L = L
38     else
39        prm%L = 8
```

```
40      end if
41      if ( present(N) ) then
42          prm%N = N
43      else
44          prm%N = 32
45      end if
46      if ( present(M) ) then
47          prm%M = M
48      else
49          prm%M = 16
50      end if
51      if ( present(Lmax) ) then
52          prm%Lmax = Lmax
53      else
54          prm%Lmax = 32
55      end if
56
57      prm%high_comm = MPI_COMM_WORLD
58      prm%low_comm = MPI_COMM_SELF
59
60
61      ncv = zpares_get_ncv(prm)
62      allocate(eigval(ncv), X(LDA, ncv), res(ncv))
63
64      call zpares_dmpssygv(prm, LDA, rowA, colA, valA, rowB, colB, valB &
65            , emin, emax, num_ev, eigval, X, res, info)
66      call zpares_finalize(prm)
67
68
69
70
71      return
72    end subroutine zpares_dmpssygv_sub
73
74    subroutine make_crs(LDA,sub_ijv,row,col,val)
75      implicit none
76      interface
77          subroutine sub_ijv(i,j,v)
78            integer,intent(in)::i,j
79            real(8),intent(out)::v
80          end subroutine sub_ijv
81      end interface
82      integer,intent(in)::LDA
83      integer,allocatable,intent(out)::row(:),col(:)
84      real(8),allocatable,intent(out)::val(:)
85  !,local, ,variables
86      integer::i,j
87      real(8)::vec_temp(1:LDA)
88      integer::vec_coltemp(1:LDA)
89      integer,allocatable::col_temp(:)
90      real(8),allocatable::val_temp(:)
91      real(8)::v
92      integer::Mi,M
93  !, ,The, ,matrix, ,must, ,be, ,symmetric
94
95      allocate(row(1:LDA+1))
96      M = 0
97      do i = 1,LDA
98          vec_temp = 0d0
99          Mi = 0
100         if(i == 1) then
101             row(i) = 1
102         else
103             row(i) = M + 1
104         end if
105         do j = i,LDA
106             call sub_ijv(i,j,v)
107             if(v .ne. 0d0) then
108                 Mi = Mi + 1
109                 vec_temp(Mi) = v
110                 vec_coltemp(Mi) = j
111             end if
112         end do
113
```

```
114         allocate(val_temp(1:M+Mi),col_temp(1:M+Mi))
115         if(i .ne. 1) then
116           val_temp(1:M) = val(1:M)
117           col_temp(1:M) = col(1:M)
118           deallocate(val,col)
119         end if
120         val_temp(M+1:M+Mi) = vec_temp(1:Mi)
121         col_temp(M+1:M+Mi) = vec_coltemp(1:Mi)
122
123         M = M + Mi
124         allocate(val(1:M),col(1:M))
125         val = val_temp
126         col = col_temp
127         deallocate(val_temp,col_temp)
128
129       end do
130       row(LDA+1) = M+1
131
132       return
133     end subroutine make_crs
134   end module zpares_subCRS
135
136
137   program main
138     use zpares_subCRS
139     implicit none
140     include 'mpif.h'
141     integer, parameter :: LDA = 5000
142     integer :: num_ev,i
143     real(8) :: emin, emax
144     integer, allocatable :: rowA(:), colA(:)
145     real(8), allocatable :: res(:), eigval(:)
146     real(8), allocatable :: valA(:), X(:,:)
147     integer::ierr
148     external make_matrix
149     call MPI_INIT(ierr)
150
151     call make_crs(LDA,make_matrix,rowA,colA,valA)
152     emin = 1.49d0
153     emax = 2.01d0
154
155     call zpares_dmpssygv_sub(LDA, rowA, colA, valA &
156           , emin, emax, num_ev, eigval, X, res)
157
158     do i = 1, num_ev
159       write(*,*) i, eigval(i), res(i)
160     end do
161
162     call MPI_FINALIZE(ierr)
163   end program main
164
165   subroutine make_matrix(i,j,v)
166     implicit none
167     integer,intent(in)::i,j
168     real(8),intent(out)::v
169     v = 0d0
170
171     if(i ==j) then
172       v = 2d0
173     else if(abs(i-j) == 1) then
174       v = 1d0
175     end if
176
177     return
178   end subroutine make_matrix
```

6 最適な設定について

通常の固有値問題を解く時は、

```
zpares_dmpssygv(prm, mat_size, rowA, colA, valA, rowB, colB, valB &
```

```
      , emin, emax, num_ev, eigval, X, res, info)
```

ではなく

```
zpares_dmpssyev(prm, mat_size, rowA, colA, valA, emin, emax, num_ev, eigval, X, res, info)
```

を使うとよい。また、

```
prm%asp_ratio = 0.2d0
```

としておくと、楕円が細長くなって精度が上がる。