4 倍精度
固有値計算ライブラリ

QPEigen Ver.1.0

ユーザーズマニュアル

2015 年 2 月

独立行政法人

日本原子力研究開発機構

目次

1 概説

原子力研究開発機構システム計算科学センターシミュレーション技術開発室では、第二期中期計画に定めた研究開発テーマ「原子炉構造材料における劣化現象の解明、燃料関連アクチノイド化合物の物質特性の予測並びに高効率な熱電材料、電源材料及び超伝導材料の構造と機能の関係解明のための高精度シミュレーション技術を開発する」を達成するため、関係する計算手法の研究開発を行っている。上記計画では、材料物性を研究する数値計算手法の開発が共通の計算科学上の課題であり、量子力学（量子多体問題）に基づき正確に電子状態を計算する手法の開発が不可欠である。一般に、量子多体問題を正確に解くためには、極めて大きな自由度を近似なしに扱う必要性があることから、大規模な数値計算を、精度を維持しながら実施しなければならない。現在、量子多体問題を解くような大規模計算は、並列計算機を利用することで実現可能であり、プロセッサー毎の利用可能メモリを集めることで全体として巨大なメモリ領域が確保されれば、計算資源が許す限りの大規模問題にアプローチできる。しかし、実際に大規模計算を行い、正確な量子状態を得るためには、膨大な数の演算が必要となるため、累積誤差が蓄積する事が知られてきた（計算機では、ある有効桁数での演算を行うため、一回の演算毎に丸め誤差が混入し、演算量が増加するほどその累積誤差は増大する)。

これまで従来規模のシミュレーションではその誤差が問題視されることはなかったが、京コンピュータのような超大規模計算機の性能を極限まで利用したシミュレーションを行うと、累積誤差は膨大になり、シミュレーション結果の有効桁がほとんどなくなる可能性があることを指摘してきた経緯がある(実際、前地球シミュレータの全ノード 4096CPU を用いて計算可能な固有値問題を解いた際、有効精度が数桁程度になることを観測している)。

このような状況の下、計算機シミュレーションにおいて頻繁に利用される基本演算ルーチンの１つである固有値計算ルーチンの 4 倍精度への拡張は、来る高精度な超並列大規模シミュレーションを実現するにあたり必須な研究開発項目であり、全ての大規模シミュレーションに共通な普遍的問題であるため、その研究開発に対する緊急性は極めて高いと判断できる。本開発に先立ち実施した基本線形代数プログラム群(BLAS)の 4 倍精度化において、既にその有効性を確認している。

我々はこれまでに実対称行列およびエルミート行列の固有値と固有ベクトルを求めるライブラリ倍精度版の固有値計算ライブラリ EigenK を開発している。この EigenK ライブラリでは実対称行列の固有値固有ベクトルを求める場合には、3 重対角化あるいは 5 重対角化を選択することができる。今回開発した 4 倍精度固有値計算ライブラリ（以下、QPEigen_s ライブラリと呼ぶ）は、この EigenK ライブラリのうち、実対称行列の固有値と固有ベクトルを 3 重対角化で求めるルーチン群を 4 倍精度化したものである。

本書では、QPEigen_s ライブラリの使用方法について説明する。

2 行列対角化について

行列の対角化とは固有値計算とほぼ同義で、構造解析や量子力学の計算でよく表れる。固有値計算とは、行列$\mathbf{A}$が与えられたときに$\mathbf{Ax} = \lambda\mathbf{x}$なる固有値$\lambda$および固有ベクトル$\mathbf{x}$を求める問題である。QPEigen_s は実対称行列の固有値を 3 重対角化を経由して求めるライブラリである。

3 4 倍精度化アルゴリズムについて

IEEE 754 規格に基づく 4 倍精度(128 bit)の浮動小数点演算は、最新の Fortran を含む一部の言語ではすでに実装されている。しかし、ハードウェアがこの演算に対してまだ最適化されていないため、この演算は任意精度演算として実行される場合が多い。その結果、演算速度が倍精度演算と比べて非常に遅くなるという課題を抱えている。

QPEigen_s では、計算精度と計算速度を両立させるために、D.H.Bailey が提案した double-double アルゴリズムを採用した。このアルゴリズムでは、2 つの倍精度データを利用して、高精度の 1 つのデータを表現し、また倍精度データの演算を組み合わせて 4 倍精度の演算を実現する。この double-double アルゴリズムを用いた数値表現の範囲や精度は、本来の 4 倍精度のそれよりは若干劣るものの、通常の倍精度数値表現と比べると約 2 倍である。このような高精度なデータに対する演算を倍精度演算のみで行えるため、大規模で高精度なシミュレーションの実現のために非常に有効なアルゴリズムである。

4 参考文献

BaileyHD. High-Precision Software Directory. http://crd-legacy.lbl.gov/~dhbailey/mpdist.
山田進, 佐々成正, 今村俊幸, , 町田昌彦. “4 倍精度基本線形代数ルーチン QPBLAS の紹介とアプリケーションへの応用.” 2012-HPC [情報処理学会研究報告] 137, 第 23 [2012]: 1-6.

5 ディレクトリ構成

メディアのディレクトリ構成を以下に示す。
（QPEigen_s の例であるが、他のライブライリも、_s を_sx や_h に変えた同様の内容を持つ）

ディレクトリ名	説明
ddmpi	4 倍精度(DD)MPI 通信ライブラリのソースファイル等一式を格納
ddblas	DDBLAS のソースファイル等一式を格納
ddlapack	DDLAPACK のソースファイル等一式を格納
ddscalapack	DDScaLAPACK のソースファイル等一式を格納
ddeigen_s	DDEigen_s のソースファイル等一式を格納
ddeigen_s_test	4 倍精度(DD)のテストを実行するためのプログラム一式を格納
eigen_s	倍精度 Eigen_s のソースファイル等一式を格納
eigen_s_test	倍精度 Eigen_s のテストを実行するためのプログラム一式を格納

6 必要なソフトウェア

QPEigen ライブラリでは、DDBLAS 及び BLAS、4 倍精度 ScaLAPACK（以下 DDScaLAPACK)、オリジナルの ScaLAPACK、4 倍精度 LAPACK（以下 DDLAPACK)、オリジナルの LAPACK、4 倍精度 MPI 通信ライブラリを使用している。

このため、リンク時（ロードモジュール作成時）に、DDBLAS 及び DDScaLAPACK、DDLAPACK をリンクするためのオプションが必要になる。これらのオプションは、configure のオプションで指定する。

7 インストール方法

以下のコマンドを端末に入力することで、ユーザーの環境に合わせた Makefile の作成とコンパイルが行われる。
（QPEigen_s の例であるが、他のライブライリも、_s を_sx や_h に変えた同様の操作を行う）

```
tar xzf QPEigen_s-version.tar.gz
cd QPEigen_s-version
./configure options
make
```

configure の際に考慮すべき主なオプションとその意味は、以下の通りである。

FC=*f90_compiler*	Fortran コンパイラの指定
CC=*c_compiler*	C コンパイラの指定
FCFLAGS=*options*	Fortran のコンパイルオプションの指定 並列計算や最適化に関する指定を行う。
CFLAGS=*options*	C のコンパイルオプションの指定
LDFLAGS=*options*	リンクオプションの指定
--prefix=*path*	インストールディレクトリの変更
--disable-openmp	OpenMP による並列化を無効にする 指定されなかった場合は、OpenMP による並列化を行う

コンパイルのために最低限必要な configure オプションは、たとえば次のようになる。

例：GNU C / GNU Fortran を利用する場合

```
 ./configure CC=mpicc FC=mpif90
```

例：Intel C / Intel Fortran を利用する場合

```
./configure CC=mpiicc FC=mpiifort CFLAGS="-O2" ¥
--disable-openmp LIBS="-L/usr/lib/gcc -lgfortran"
```

make コマンドのオプションは以下のものがある

make	単に make することは、make all と同義。
make all	すべてのソースコードをコンパイルする。
make lib	ライブラリのみをコンパイルする。
make test	テストプログラムのみをコンパイルする。 ただし、もし lib フォルダに依存ライブラリ(ddblas, blas, lapack, scalapack)が存在しない場合にはエラーを出力して停止する。 また、その他の同梱している依存ライブラリ(ddmpi, ddlapack, ddscalapack, ddeigen_s …)が見つからない場合には自動的に make lib される。
make sample	サンプルプログラムをコンパイルする。
make bench	ベンチマークプログラムのみをコンパイルする。
make verify	検証プログラムをコンパイルする。
make eigen_s_test	テストプログラムのみをコンパイルする。
make clean	全てのコンパイル済ファイルを削除する。
make clean-lib	ライブラリに関する全コンパイル済ファイルを削除する。
make clean-test	テストプログラム関係のみ削除する。
make clean-[programs]	個別にライブラリ関係のファイルを削除する。

make が成功すると、次のライブラリやプログラムが作成される。

ddlapack/libddlapack.a	DDLAPACK ライブラリ
ddmpi/libMpifdd1.a	DDMPI ライブラリ
ddmpi/libMpifdd2.a	DDMPI ライブラリ
ddscalapack/libddscalapack.a	DDScaLAPACK ライブラリ
eigen_s/libEigen_s.a	倍精度 Eigen ライブラリ
ddeigen_s/libddEigen_s.a	double-double 4 倍精度 Eigen ライブラリ
eigen_s_test/test_frank_d	倍精度フランク行列固有値検証プログラム
eigen_s_test/test_random_d	倍精度ランダム行列固有値検証プログラム
ddeigen_s_test/test_frank_d	double-double 4 倍精度フランク行列固有値検証プログラム
ddeigen_s_test/test_random_d	double-double 4 倍精度ランダム行列固有値検証プログラム
sample/frank5_sample_dd	5 次フランク行列の固有値を求めるサンプルプログラム

sample/hilbert5_sample_dd	5 次ヒルベルト行列の固有値を求めるサンプルプログラム
verify/ddeigen_s_verify	実対称行列の固有値と固有ベクトルを求める検証用プログラム
bench/bench_frank_matrix	フランク行列用の性能測定プログラム
bench/bench_randam_matrix	ランダム行列用の性能測定プログラム

8 検証用・性能評価用プログラム

7 に示した手順によって、コンパイルを完了させたのち、次のコマンドを入力する。
（QPEigen_s の例であるが、他のライブライリも、_s を_sx や_h に変えた同様の操作を行う）

```
cd ddeigen_s_test
./test_frank_dd （フランク行列の場合）
./test_random_dd （ランダム行列の場合）
```

テストプログラムを実行すると標準出力に計算精度が出力される。この計算精度で 4 倍精度の演算が行われているかを判断する。計算精度は以下のように計算されている。

計算精度 ＝ $\max\left|Ax_n - w_n x_n\right|$

w_n ：数値解法により求めた固有値
A ：精度検証行列
x_n ：数値解法により求めた固有ベクトル

図 8-1 に標準出力の例を示す。赤枠部分に計算精度が出力される。

図 8-1 テストプログラム実行後の標準出力の例

```
----- Check a calculation result. -----
max|Ax-wx|=  9.276954941592523E-027  0.000000000000000E+000        100
sum|Ax-wx|=  6.764819935969023E-026  2.788776773448148E-042
|A|=   5050.00000000000       0.000000000000000E+000
epsilon=  2.220446049250313E-016  0.000000000000000E+000
max|Ax-wx|/Ne|A|=  8.273206102591875E-017
|ZZ-I|=  1.104682272602909E-029  1.220322923403127E-058
---------------------------------------
```

各ルーチンの呼び出し方法や引数の説明は 0 を参照のこと。

9 サンプルプログラム

sample ディレクトリに Fortran77 で書かれた QPEigen_s のサンプルプログラムがいくつか用意されている。

（QPEigen_s の例であるが、他のライブライリも、_s を_sx や_h に変えた同様の操作を行う）

frank5_sample_dd	5 次のフランク行列の対角化を double-double4 倍精度で行う
hilbert5_sample_dd	5 次のヒルベルト行列の対角化を double-double4 倍精度で行う

9.1 フランク行列の対角化（frank5_sample_dd）

フランク行列とは、固有値が既知の条件数の悪い行列として、固有値を数値計算する場合のベンチマークによく用いられる。5 次の場合は次のような行列である。

$$\begin{bmatrix} 5 & 4 & 3 & 2 & 1 \\ 4 & 4 & 3 & 2 & 1 \\ 3 & 3 & 3 & 2 & 1 \\ 2 & 2 & 2 & 2 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

4 倍精度の行列$A(=Ah+Al)$に、フランク行列の成分を high word に代入し、low word は 0.0d とする。あらかじめ必要な作業領域の大きさを cstab_get_optdim サブルーチンを使って求めておく。ddmatrix_adjust_s サブルーチンで、行列を作業領域にコピーしたあとで、ddeigen_s サブルーチンを呼び出すと、固有値および固有ベクトルが得られる。

9.2 ヒルベルト行列の対角化（hilbert5_sample_dd）

ヒルベルト行列は悪条件の行列の代表として数値計算のベンチマークによく用いられる。5 次の場合は次のような行列である。

$$\begin{bmatrix} 1 & \frac{1}{2} & \frac{1}{3} & \frac{1}{4} & \frac{1}{5} \\ \frac{1}{2} & \frac{1}{3} & \frac{1}{4} & \frac{1}{5} & \frac{1}{6} \\ \frac{1}{3} & \frac{1}{4} & \frac{1}{5} & \frac{1}{6} & \frac{1}{7} \\ \frac{1}{4} & \frac{1}{5} & \frac{1}{6} & \frac{1}{7} & \frac{1}{8} \\ \frac{1}{5} & \frac{1}{6} & \frac{1}{7} & \frac{1}{8} & \frac{1}{9} \end{bmatrix}$$

4 倍精度の行列$A(=Ah+Al)$に、ヒルベルト行列の成分を 4 倍精度の割り算の結果として high word および low word に代入する。あらかじめ必要な作業領域の大きさを cstab_get_optdim サブルーチンを使って求めておく。ddmatrix_adjust_s サブルーチンで、行列を作業領域にコピーしたあとで、ddeigen_s サブルーチンを呼び出すと、固有値および

固有ベクトルが得られる。

9.3 サンプルプログラムの実行方法

これらのサンプルプログラムは make の際にコンパイルされているため、例えば frank5_sample_dd の場合は次のように入力することで実行できる。

```
cd sample
./frank5_sample_dd
```

このプログラムを自分でコンパイルする場合は、例えば次のように行う。

```
mpiifort frank5_sample_dd.f -I../ddeigen_s ¥
 ../ddeigen_s/libddEigen_s.a ../ddscalapack/libddscalapack.a ¥
 ../ddlapack/libddlapack.a ¥
 ../ddmpi/libMpifdd1.a ../ddmpi/libMpifdd2.a ¥
 ../../lib/libdddblas.a ../../lib/libscalapack.a ¥
 ../../lib/liblapack.a ../../lib/libblas.a
```

サンプルプログラム frank5_sample_dd.f のソースコードは以下の通り。

```
      program frank5_sample_dd
      use ddcommunication_s, only : eigen_init
     &                      , eigen_free
      implicit double precision (a-h,o-v,x-z)

      double precision, allocatable :: ah(:,:), al(:,:)
      double precision, pointer :: bh(:), bl(:)
      double precision, pointer :: zh(:), zl(:)
      double precision, pointer :: wh(:), wl(:)
      logical iexist
!
      include 'mpif.h'
      include 'trd.h'
!
      call mpi_init(ierr)
      call mpi_comm_rank(mpi_comm_world,i$inod,ierr)
```

```
      call mpi_comm_size(mpi_comm_world,i$nnod,ierr)

      n=5

      allocate(ah(n,n),al(n,n))

      call eigen_init(2)
      NPROW = size_of_col
      NPCOL = size_of_row
      nx = ((n-1)/NPROW+1)
      call CSTAB_get_optdim(nx, 2, 1, 2, nm)
!      call CSTAB_get_optdim(nx, 6, 16*4, 16*4*2, nm)
      call eigen_free(0)
      NB  = 32
!      NB  = 64+32
      nmz = ((n-1)/NPROW+1)
      nmz = ((nmz-1)/NB+1)*NB+1
      nmw = ((n-1)/NPCOL+1)
      nmw = ((nmw-1)/NB+1)*NB+1

      larray = MAX(nmz,nm)*nmw

      allocate( bh(larray), bl(larray),zh(larray),zl(larray),
     &        wh(n),wl(n), stat=istat)
      if(istat.ne.0) then
         print*,"Memory exhausted"
         call flush(6)
         stop
      endif

      ah(1,1:5) = (/ 5.0d0, 4.0d0, 3.0d0, 2.0d0, 1.0d0 /)
      ah(2,1:5) = (/ 4.0d0, 4.0d0, 3.0d0, 2.0d0, 1.0d0 /)
      ah(3,1:5) = (/ 3.0d0, 3.0d0, 3.0d0, 2.0d0, 1.0d0 /)
      ah(4,1:5) = (/ 2.0d0, 2.0d0, 2.0d0, 2.0d0, 1.0d0 /)
      ah(5,1:5) = (/ 1.0d0, 1.0d0, 1.0d0, 1.0d0, 1.0d0 /)
```

```
      al(1,1:5) = (/ 0.0d0, 0.0d0, 0.0d0, 0.0d0, 0.0d0 /)
      al(2,1:5) = (/ 0.0d0, 0.0d0, 0.0d0, 0.0d0, 0.0d0 /)
      al(3,1:5) = (/ 0.0d0, 0.0d0, 0.0d0, 0.0d0, 0.0d0 /)
      al(4,1:5) = (/ 0.0d0, 0.0d0, 0.0d0, 0.0d0, 0.0d0 /)
      al(5,1:5) = (/ 0.0d0, 0.0d0, 0.0d0, 0.0d0, 0.0d0 /)

      call ddmatrix_adjust_s(n, ah, al, bh, bl, nm)

      call ddeigen_s(n, bh, bl, nm, wh, wl, zh, zl, nm, m, 1)

      write(*,*) 'EigenValue=', wh

      deallocate(ah,al)
      deallocate(bh,bl)
      deallocate(zh,zl)
      deallocate(wh,wl)
*
      call MPI_Finalize(ierr)
      end
```

10 サブルーチン詳細

QPEigen_s ライブラリのルーチン・モジュールの一覧を表 1 に示す。

表 1

ルーチン名	機能
ddeigen_s	実対称行列を HouseHolder3 重対角化して固有値及び固有ベクトルを求める。3 重対角化・固有値算出・固有ベクトル算出の一連の動作を実行する。
ddeigen_trd	実対称行列を HouseHolder3 重対角化により 3 重対角化する。
ddeigen_dc	実対称行列の HouseHolder3 重対角化により、3 重対角化された行列を入力とし、分割統治法で固有値を求める。
ddeigen_tbk	実対称行列の HouseHolder3 重対角化に対応する固有ベクトルを求める。
ddmatrix_adjust_s	DDEigen ライブラリでは、2 次元のサイクリック分割を行っている。行列のデータ配置をサイクリック分割に対応させるための変換

	ルーチンである。当ライブラリを使用する直前に呼び出す必要がある。
ddcommunication_s	MPI 通信を行うためのモジュール

QPEigen_sx ライブラリのルーチン・モジュールの一覧を表 12 に示す。

表 2

ルーチン名	機能
ddeigen_sx	実対称行列を NarrowBandReduction 5 重対角化して固有値及び固有ベクトルを求める。5 重対角化・固有値算出・固有ベクトル算出の一連の動作を実行する。
ddeigen_prd	実対称行列を NarrowBandReduction 5 重対角化により 5 重対角化する。
ddeigen_dcx	実対称行列の NarrowBandReduction 5 重対角化により、5 重対角化された行列を入力とし、分割統治法で固有値を求める。
ddeigen_pbk	実対称行列の NarrowBandReduction 5 重対角化に対応する固有ベクトルを求める。
ddmatrix_adjust_sx	DDEigen ライブラリでは、2 次元のサイクリック分割を行っている。行列のデータ配置をサイクリック分割に対応させるための変換ルーチンである。当ライブラリを使用する直前に呼び出す必要がある。
ddcommunication_sx	MPI 通信を行うためのモジュール

QPEigen_h ライブラリのルーチン・モジュールの一覧を表 13 に示す。

表 3

ルーチン名	機能
ddeigen_h	エルミート行列の固有値固有ベクトルを求める。対角化、固有値計算、固有ベクトル計算の一連の動作を実行する。
ddeigen_hrd	エルミート行列を対角化する。
ddeigen_dch	エルミート行列について、対角化された行列を入力とし、分割統治法で固有値を求める。
ddeigen_hbk	エルミート行列に対応した固有ベクトルを求める。
ddmatrix_adjust_h ddmatrix_adjust_hi	DDEigen ライブラリでは、2 次元のサイクリック分割を行っている。行列のデータ配置をサイクリック分割に対応させるための変換ルーチンである。当ライブラリを使用する直前に呼び出す必要がある。
ddcommunication_h	MPI 通信を行うためのモジュール

ルーチン名	ddeigen_s			
概要	実対象行列を HouseHolder-3 重対角化して固有値及び固有ベクトルを求める。3 重対角化、固有値及び固有ベクトルの導出の一連の動作を実行。			
呼び出し方法	ddeigen_s(n, ah, al, lda, wh, wl, zh, zl, ldz, m, ifl)			
引数	説明		i/o	備考
n	行列の次元	integer	i	
ah	行列を格納した配列の High-word	double	i	
al	行列を格納した配列の Low-word	double	i	
lda	配列 a のサイズ	integer	i	
wh	固有値を格納する配列の High-word	double	o	
wl	固有値を格納する配列の Low-word	double	o	
zh	固有ベクトルを格納する配列の High-word	double	o	
zl	固有ベクトルを格納する配列の Low-word	double	o	
ldz	配列 z のサイズ	integer	i	
m	ブロック化係数	integer	i	32～128 程度
ifl	固有ベクトルの要否フラグ	integer	i	0 または 1

ルーチン名	ddeigen_trd			
概要	実対称行列を HouseHolder-3 重対角化により 3 重対角化する			
呼び出し方法	ddeigen_trd(n, ah, al, lda, dh, dl, eh, el, m)			
引数	説明		i/o	備考
n	行列の次元	integer	i	
ah	行列を格納した配列の High-word	double	i	
al	行列を格納した配列の Low-word	double	i	
lda	配列 a のサイズ	integer	i	
dh	対角要素を格納する配列の High-word	double	o	
dl	対角要素を格納する配列の Low-word	double	o	
eh	副対角要素を格納する配列の High-word	double	o	
el	副対角要素を格納する配列の Low-word	double	o	
m	ブロック化係数	integer	i	

ルーチン名	ddeigen_dc			
概要	実対象行列の HouseHolder-3 重対角化により、3 重対角化された行列を入力とし、分割統治法で固有値を求める			
呼び出し方法	ddeigen_dc(n, wh, wl, eh, el, zh, zl, ldz, info)			
引数	説明		i/o	備考
n	行列の次元	integer	i	
wh	固有値を格納する配列の High-word	double	o	
wl	固有値を格納する配列の Low-word	double	o	
eh	副対角要素を格納する配列の High-word	double	i	
el	副対角要素を格納する配列の Low-word	double	i	
zh	固有ベクトルを格納する配列の High-word	double	o	
zl	固有ベクトルを格納する配列の Low-word	double	o	
ldz	配列 z のサイズ	integer	i	
info	エラー番号	integer	o	

ルーチン名	ddeigen_tbk			
概要	実対称行列の HouseHolder-3 重対角化に対応する固有ベクトルを求める			
呼び出し方法	ddeigen_tbk(n, ah, al, lda, zh, zl, ldz, eh, el, m)			
引数	説明		i/o	備考
n	行列の次元	integer	i	
ah	行列を格納した配列の High-word	double	i	
al	行列を格納した配列の Low-word	double	i	
lda	配列 a のサイズ	integer	i	
zh	固有ベクトルを格納する配列の High-word	double	o	
zl	固有ベクトルを格納する配列の Low-word	double	o	
ldz	配列 z のサイズ	integer	i	
eh	副対角要素を格納する配列の High-word	double	i	
el	副対角要素を格納する配列の Low-word	double	i	
m	ブロック化係数	integer	i	128 程度を指定

ルーチン名	ddmatrix_adjust_s			
概要	DDEigen ライブラリでは、2 次元のサイクリック分割を行っている。行列のデータ配置をサイクリック分割に対応させるための変換ルーチンである。当ライブラリを使用する直前に呼び出す必要がある。			
呼び出し方法	ddmatrix_adjust_s(n, ah, al, bh, bl, nm)			
引数	説明		i/o	備考
n	行列の次元	integer	i	
ah	行列を格納した配列の High-word	double	i	
al	行列を格納した配列の Low-word	double	i	
bh	分散データ行列の格納配列の High-word	integer	o	
bl	分散データ行列の格納配列の Low-word	double	o	
nm	配列 b のサイズ	integer	i	

ルーチン名	ddeigen_sx			
概要	実対称行列を NarrowBandReduction 5 重対角化して固有値及び固有ベクトルを求める。5 重対角化・固有値算出・固有ベクトル算出の一連の動作を実行する。			
呼び出し方法	ddeigen_sx(n, ah, al, nma, wh, wl, zh, zl, dh, dl, eh, el, nme, m0, ifl)			
引数	説明		i/o	備考
n	行列の次元	integer	i	
ah	行列を格納した配列の High-word	double	i	
al	行列を格納した配列の Low-word	double	i	
nma	配列 a のサイズ	integer	i	
wh	固有値を格納する配列の High-word	double	o	
wl	固有値を格納する配列の Low-word	double	o	
zh	固有ベクトルを格納する配列の High-word	double	o	
zl	固有ベクトルを格納する配列の Low-word	double	o	
dh	対角要素を格納する配列の High-word		o	
dl	対角要素を格納する配列の Low-word		o	
eh	副対角要素を格納する配列の High-word		o	
el	副対角要素を格納する配列の Low-word		o	
nme	配列 e のサイズ	integer	i	
m0	ブロック化係数	integer	i	32～128 程度
ifl	固有ベクトルの要否フラグ	integer	i	0 または 1

ルーチン名	ddeigen_prd			
概要	実対称行列を NarrowBandReduction 5 重対角化により 5 重対角化する			
呼び出し方法	ddeigen_prd(n, ah, al, nma, dh, dl, eh, el, nme, m0)			
引数	説明		i/o	備考
n	行列の次元	integer	i	
ah	行列を格納した配列の High-word	double	i	
al	行列を格納した配列の Low-word	double	i	
nma	配列 a のサイズ	integer	i	
dh	対角要素を格納する配列の High-word	double	o	
dl	対角要素を格納する配列の Low-word	double	o	
eh	副対角要素を格納する配列の High-word	double	o	
el	副対角要素を格納する配列の Low-word	double	o	
nme	配列 e のサイズ	integer	i	
m0	ブロック化係数	integer	i	32～128 程度

ルーチン名	ddeigen_dcx			
概要	実対称行列の NarrowBandReduction 5 重対角化により、5 重対角化された行列を入力とし、分割統治法で固有値を求める。			
呼び出し方法	ddeigen_dcx(n, wh, wl, eh, el, nme, zh, zl, nmz)			
引数	説明		i/o	備考
n	行列の次元	integer	i	
wh	固有値を格納する配列の High-word	double	o	
wl	固有値を格納する配列の Low-word	double	o	
eh	副対角要素を格納する配列の High-word	double	o	
el	副対角要素を格納する配列の Low-word	double	o	
nme	配列 e のサイズ	integer	i	
zh	固有ベクトルを格納する配列の High-word	double	o	
zl	固有ベクトルを格納する配列の Low-word	double	o	
nmz	配列 z のサイズ	integer	i	

ルーチン名	ddeigen_pbk			
概要	実対称行列の NarrowBandReduction 5 重対角化に対応する固有ベクトルを求める。			
呼び出し方法	ddeigen_pbk(n, ah, al, nma, zh, zl, nmz, eh, el, m0)			
引数	説明		i/o	備考
n	行列の次元	integer	i	
ah	行列を格納した配列の High-word	double	i	
al	行列を格納した配列の Low-word	double	i	
nma	行列 a のサイズ	integer	i	
lda	配列 a のサイズ	integer	i	
zh	固有ベクトルを格納する配列の High-word	double	o	
z l	固有ベクトルを格納する配列の Low-word	double	o	
nmz	配列 z のサイズ	integer	i	
eh	副対角要素を格納する配列の High-word	double	i	
el	副対角要素を格納する配列の Low-word	double	i	
m	ブロック化係数	integer	i	128 程度を指定

ルーチン名	ddmatrix_adjust_sx			
概要	DDEigen ライブラリでは、2 次元のサイクリック分割を行っている。行列のデータ配置をサイクリック分割に対応させるための変換ルーチンである。当ライブラリを使用する直前に呼び出す必要がある。			
呼び出し方法	ddmatrix_adjust_sx(n, ah, al, bh, bl, nm)			
引数	説明		i/o	備考
n	行列の次元	integer	i	
ah	行列を格納した配列の High-word	double	i	
al	行列を格納した配列の Low-word	double	i	
bh	分散データ行列の格納配列の High-word	integer	o	
bl	分散データ行列の格納配列の Low-word	double	o	
nm	配列 b のサイズ	integer	i	

ルーチン名	ddeigen_h			
概要	エルミート行列の固有値固有ベクトルを求める。対角化、固有値計算、固有ベクトル計算の一連の動作を実行する。			
呼び出し方法	ddeigen_h(n, arh, arl, aih, ail, wh, wl, lda, m0, ifl)			
引数	説明		i/o	備考
n	行列の次元	integer	i	
arh	行列（実部）を格納した配列の High-word (出力時) 固有ベクトル実部の High-word	double	i/o	
arl	行列（実部）を格納した配列の Low-word (出力時) 固有ベクトル実部の Low-word	double	i/o	
aih	行列（虚部）を格納した配列の High-word (出力時) 固有ベクトル虚部の High-word	double	i/o	
ail	行列（虚部）を格納した配列の Low-word (出力時) 固有ベクトル虚部の Low-word	double	i/o	
wh	固有値を格納する配列の High-word	double	o	
wl	固有値を格納する配列の Low-word	double	o	
lda	配列 a のサイズ	integer	i	
m0	ブロック化係数	integer	i	32～128 程度
ifl	固有ベクトルの要否フラグ	integer	i	0 または 1

ルーチン名	ddeigen_hrd			
概要	エルミート行列を対角化する。			
呼び出し方法	ddeigen_hrd(n, arh, arl, aih, ail, lda, dh, dl, eh, el, e2h, e2l, tauh, taul, m, zrh, zrl, zih, zil)			
引数	説明		i/o	備考
n	行列の次元	integer	i	
arh	行列（実部）を格納した配列の High-word	double	i	
arl	行列（実部）を格納した配列の Low-word	double	i	
aih	行列（虚部）を格納した配列の High-word	double	i	
ail	行列（虚部）を格納した配列の Low-word	double	i	
lda	配列 a のサイズ	integer	i	
dh	対角要素を格納する配列の High-word	double	o	
dl	対角要素を格納する配列の Low-word	double	o	
eh	副対角要素を格納する配列の High-word	double	o	
el	副対角要素を格納する配列の Low-word	double	o	
e2h	ワーク領域	double	o	
e2l	ワーク領域	double	o	
tauh	ワーク領域	double	o	
taul	ワーク領域	double	o	
m	ブロック化係数	integer	i	128 程度
zrh	固有ベクトル（実部）の High-word	double	o	
zrl	固有ベクトル（実部）の Low-word	double	o	
zih	固有ベクトル（虚部）の High-word	double	o	
zil	固有ベクトル（虚部）の Low-word	double	o	

ルーチン名	ddeigen_dch			
概要	エルミート行列について、対角化された行列を入力とし、分割統治法で固有値を求める。			
呼び出し方法	ddeigen_dch(n, dh, dl, eh, el, zh, zl, ldz, info)			
引数	説明		i/o	備考
n	行列の次元	integer	i	
dh	固有値を格納する配列の High-word	double	o	
dl	固有値を格納する配列の Low-word	double	o	
eh	副対角要素を格納する配列の High-word	double	o	
el	副対角要素を格納する配列の Low-word	double	o	
zh	固有ベクトル（実部）を格納する配列の High-word	double	o	
zl	固有ベクトル（実部）を格納する配列の Low-word	double	o	
ldz	配列 z のサイズ	integer	i	
info	エラー番号	integer	o	

ルーチン名	ddeigen_hbk			
概要	エルミート行列に対応した固有ベクトルを求める。			
呼び出し方法	ddeigen_hbk(arh, arl, aih, ail, lda, zrh, zrl, zih, zil, ldz, tauh, taul, ltau, n)			
引数	説明		i/o	備考
arh	行列（実部）を格納した配列の High-word	double	i	
arl	行列（実部）を格納した配列の Low-word	double	i	
aih	行列（虚部）を格納した配列の High-word	double	i	
ail	行列（虚部）を格納した配列の Low-word	double	i	
lda	行列 a のサイズ	integer	i	
zrh	固有ベクトル（実部）を格納する配列の High-word	double	o	
zrl	固有ベクトル（実部）を格納する配列の Low-word	double	o	
zih	固有ベクトル（虚部）を格納する配列の High-word	double	o	
zil	固有ベクトル（虚部）を格納する配列の Low-word	double	o	
ldz	配列 z のサイズ	integer	i	
touh	ワークエリア	double	i	
toul	ワークエリア	double	i	
ltau	ワークエリア	integer	i	
n	行列 a の次元	integer	i	

ルーチン名	ddmatrix_adjust_h, ddmatrix_adjust_hi			
概要	DDEigen ライブラリでは、2 次元のサイクリック分割を行っている。行列のデータ配置をサイクリック分割に対応させるための変換ルーチンである。当ライブラリを使用する直前に呼び出す必要がある。			
呼び出し方法	ddmatrix_adjust_h(n, ah, al, bh, bl, nm) ddmatrix_adjust_hi(n, ah, al, bh, bl, nm)			
引数	説明		i/o	備考
n	行列の次元	integer	i	
ah	行列を格納した配列の High-word	double	i	

al	行列を格納した配列の Low-word	double	i	
bh	分散データ行列の格納配列の High-word	integer	o	
bl	分散データ行列の格納配列の Low-word	double	o	
nm	配列 b のサイズ	integer	i	