Quadruple Precision Eigenvalue Calculation Library

QPEigen Ver.1.0

User's Manual

Feb, 2015

Japan Atomic Energy Agency

Contents

1 Overview

The R&D Office of Simulation Technology at the Center for Computational Science & e-Systems in the Japan Atomic Energy Agency has been engaged in research and development (R&D) for computational techniques relevant to one of the R&D targets in its Mid-Term Plan for the second mid-term, "developing high-precision simulation technologies that can be used for clarification of degradation phenomena of the structural materials of a nuclear reactor, prediction of the material properties of fuel-related actinide compounds, and clarification of the relationship between the structures and functions of high efficiency thermoelectric materials, electric power supply materials, and superconducting materials." In the aforementioned plan, it is a common challenge of computational science to develop numerical calculation techniques for use in the study of material properties, and it is indispensable to develop methods to accurately calculate an electronic state based on quantum mechanics (quantum many-body problem). In order to accurately solve a quantum many-body problem, a very large degree of freedom must be handled without approximation; therefore, it is necessary to perform large-scale numerical calculations while maintaining sufficient precision. At present, large-scale computation, such as solving a quantum many-body problem, is made possible by using a parallel computer. As long as the total amount of the available memory space collected from each processor is sufficiently large, a problem as large as the computational resources allow is approachable. In contrast, a significant amount of computational error accumulates during an actual large-scale calculation for obtaining accurate quantum states because an enormous number of operations is required to do so. (A computer calculates with a certain number of significant digits. In every calculation, a rounding error is added. The rounding errors accumulate as the number of calculations increases.)

Until recently, the scale of simulations was not sufficiently large to be problematic in terms of error, but it has been pointed out that, when a simulation that requires a very large scale computer such as the K computer to run at its maximum performance is conducted, the rounding errors may add up significantly, and the simulation result may lose most of its significant digits. (It was actually observed that the result of an eigenvalue problem that took all 4096 CPU nodes of a previous version of the Earth Simulator to calculate had only a couple of significant digits.)

Under the situation described above, extending an eigenvalue calculation routine, one of the routines that perform basic operations and are used frequently in computer

simulations, to quadruple precision is an indispensable R&D activity for facilitating high-precision very-large-scale parallel simulations in the future. The priority of this R&D activity can be judged as very high because this problem is universal and applicable to all large-scale simulations. Extending a routine to quadruple precision has already been proved effective in our project to convert Basic Linear Algebra Subprograms (BLAS) to quadruple precision, which was conducted prior to this project.
We have developed EigenK, a double precision version of an eigenvalue calculation library used for calculating the eigenvalues and eigenvectors of a real symmetric matrix and a Hermitian matrix. When the eigenvalues and eigenvectors of a symmetric matrix are calculated with the EigenK library, it is possible to select whether to use the tridiagonal matrix algorithm or the pentadiagonal matrix algorithm.
In this work, we present three libraries, which are the quadruple precision version of the EigenK library routines. The first is the quadruple precision eigenvalue calculation library, which uses the tridiagonal matrix algorithm to calculate the eigenvalues and eigenvectors of a real symmetric matrix (hereinafter referred to as "QPEigen_s library"). The second is the "QPEigen_sx library." It is essentially similar to the QPEigen_s library, except for the fact that it uses the pentadiagonal matrix algorithm. The third is the “QPEigen_h library.” This library calculates the eigenvalues and eigenvectors of a Hermitian matrix with the tridiagonal matrix algorithm.
This document describes how to use the QPEigen_s,QPEigen_sx and QPEigen_h library (hereinafter collectively referred to as QPEigen libraries).

2 Matrix diagonalization

Matrix diagonalization is essentially equivalent to eigenvalue calculation and is frequently used in structural analysis and quantum mechanics calculations. Eigenvalue calculation is a process of calculating an eigenvalue λ and eigenvector $\mathbf{x}$ that satisfy $\mathbf{Ax} = \lambda\mathbf{x}$ when a matrix $\mathbf{A}$ is given. QPEigen_s and QPEigen_sx are libraries used for calculating the eigenvalues of a real symmetric matrix. When A is a Hermitian matrix, QPEigen_h can be used.

3 Quadruple precision algorithm

IEEE 754- compliant quadruple precision (128 bit) floating point operations have already been implemented in some languages including the latest Fortran. However,

hardware has not yet been optimized for those operations, and they are often executed as arbitrary precision calculations. As a result, their calculation speed is extremely slow compared to double precision operations.
In order to improve both the precision and the speed of the calculation, QPEigen libraries use the double-double algorithm proposed by D.H. Bailey. In this algorithm, one high-precision number is represented by a combination of two double precision numbers, and a quadruple precision operation is implemented by combining double precision operations. The range and precision of the numeric representation used in the double-double algorithm is slightly worse than those of genuine quadruple precision but is still almost double that of standard double precision. This algorithm allows calculations on such high-precision numbers using just a combination of double precision operations and therefore is highly effective for implementing a large-scale high-precision simulation.

4 References

Bailey, H.D. High-Precision Software Directory.
http://crd-legacy.lbl.gov/~dhbailey/mpdist.
Susumu Yamada, Narimasa Sasa, Toshiyuki Imamura, and Masahiko Machida: "Introduction to quadruple precision basic linear algebra routines QPBLAS and its application," IPSJ SIG Technical Report, Vol. 2012-HPC-137, No. 23, pp. 1-6, 2012.

5 Directory structure

The directory structure in the medium is described below.

There are three top directories, Eigen_s-version, Eigen_sx-version, and Eigen_h-version, corresponding to each of the libraries.

Eigen_s-version has the following subdirectories. The other libraries also have similar directory structures.

Directory name	Description
ddmpi	Stores a set of quadruple precision (DD) MPI communication library files including source files.

ddblas	Stores a set of DDBLAS files including source files.
ddlapack	Stores a set of DDLAPACK files including source files.
ddscalapack	Stores a set of DDScaLAPACK files including source files.
ddeigen_s	Stores a set of DDEigen_s files including source files.
ddeigen_s_test	Stores a set of programs that run quadruple precision (DD) tests.
eigen_s	Stores a set of double precision Eigen_s files including source files.
eigen_s_test	Stores a set of programs that run double precision Eigen_s tests.
sample	Stores a set of programs that run quadruple precision (DD) sample routines.
bench	Stores a set of programs that run quadruple precision (DD) benchmark routines.
verify	Stores a set of programs that run quadruple precision (DD) verification routines.

6 Required software

The QPEigen library uses DDBLAS, BLAS, quadruple precision ScaLAPACK (hereinafter "DDScaLAPACK"), original ScaLAPACK, quadruple precision LAPACK (hereinafter "DDLAPACK"), original LAPACK, and the quadruple precision MPI communication library.
Therefore, you must specify the appropriate options to link DDBLAS, DDScaLAPACK, and DDLAPACK when you link (when load modules are created). Those options are specified in the configure command.

7 Installation

Enter the following commands to a terminal to create a Makefile that matches a user environment and then compile.
(The following examples are for QPEigen_s. For the other libraries, replace _s with the appropriate suffixes.)

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

```
Make
```

The major options you may consider using in the `configure` command and their meanings are as follows.

FC=*f90_compiler*	Specifies the Fortran compiler.
CC=*c_compiler*	Specifies the C compiler.
FCFLAGS=*options*	Specifies Fortran compile options. Specifies parallel computation and optimization settings.
CFLAGS=*options*	Specifies C compile options.
LDFLAGS=*options*	Specifies link options.
--prefix=*path*	Changes the install directory.
--disable-openmp	Disables parallel processing via OpenMP. If this option is not specified, parallel processing is enabled using OpenMP.

Examples of the minimum options of the `configure` command necessary for compile are as follows.

Example: When GNU C and GNU Fortran are used

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

Example: When Intel C and Intel Fortran are used

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

The following list shows options for the make command.

Make	Simple "make" is the same as "make all".
make all	Compiles all source codes.
make lib	Compiles only the libraries.
make test	Compiles only the test programs. Note that if the lib folder does not contain dependent libraries (ddblas, blas, lapack, and scalapack), an error is output and then stops.

	If other dependent libraries (ddmpi, ddblas, ddlapack, ddscalapack, and ddeigen_s ...), which are included in the package, cannot be found, "make lib" is automatically executed.
make sample	Compiles sample programs.
make bench	Compiles benchmark programs.
make verify	Compiles verification programs.
make eigen_s_test	Compiles only eigen_s_test.
make clean	Deletes all compiled files.
make clean-lib	Deletes all files related to libraries.
make clean-test	Deletes all files related to test programs.
make clean-[programs]	Deletes all files related to the specific programs(sample,bench verify,eigen_s_test…).

If the make is successful, the following libraries and programs are created.

ddlapack/libddlapack.a	DDLAPACK library
ddmpi/libMpifdd1.a	DDMPI library
ddmpi/libMpifdd2.a	DDMPI library
ddscalapack/libddscalapack.a	DDScaLAPACK library
eigen_s/libEigen_s.a	Double precision Eigen library
ddeigen_s/libddEigen_s.a	Double-double quadruple precision Eigen library
eigen_s_test/test_frank_d	Double precision Frank matrix eigenvalue verification program
eigen_s_test/test_random_d	Double precision random matrix eigenvalue verification program
ddeigen_s_test/test_frank_d	Double-double quadruple precision Frank matrix eigenvalue verification program
ddeigen_s_test/test_random_d	Double-double quadruple precision random matrix eigenvalue verification program
sample/frank5_sample_dd	Sample program for calculating the eigenvalues of the Frank matrix of order 5.
sample/hilbert5_sample_dd	Sample program for calculating the eigenvalues of the Hilbert matrix of order 5.
verify/ddeigen_s_verify	Verification program for calculating eigen values

	and eigen vectors of the real symmetric matrix.
bench/bench_frank_matrix	Benchmark program for calculating the eigenvalues of the Frank matrix.
bench/bench_randam_matrix	Benchmark program for calculating the eigenvalues of the Randam data matrix.

8 Verification and performance evaluation programs

After you follow the procedure in 7 and finish the compile process, enter the following commands.
(The following examples are for QPEigen_s. For the other libraries, replace _s with the appropriate suffixes)

```
cd ddeigen_s_test
./test_frank_dd  (executing the Frank matrix test)
./test_random_dd  (executing the random matrix test)
```

When you execute a test program, the accuracy of the calculation is output to the standard output. You can determine whether a calculation is performed in quadruple precision based on the accuracy of the calculation. The accuracy of the calculation is calculated as follows.

(Accuracy of calculation) = $\max\left|Ax_n - w_n x_n\right|$

w_n : Eigenvalues calculated by numerical solution
A : Matrix set up for precision verification
x_n : Eigenvectors calculated by numerical solution

Figure 8-1 shows an example of the standard output. The output of the accuracy of the calculation is shown in the red box.

```
----- 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
---------------------------------------
```

Figure 8-1 Example of Standard Output after Executing a Test Program

For information about the procedure for calling each routine and a description of the arguments, see 0.

9 Sample programs

The sample directory contains some QPEigen sample programs written in Fortran77.

frank5_sample_dd	Diagonalizing the Frank matrix of order 5 in double-double quadruple precision
hilbert5_sample_dd	Diagonalizing the Hilbert matrix of order 5 in double-double quadruple precision

9.1 Diagonalizing the Frank matrix (frank5_sample_dd)

Frank matrices are ill-conditioned matrices with known eigenvalues and are used in a benchmark test for the numerical calculation of eigenvalues. The Frank matrix of order 5 is as follows.

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

A quadruple precision matrix $A(=Ah+Al)$ is given by substituting the elements of the Frank matrix for the high word and 0.0d for the low word. The cstab_get_optdim subroutine is called beforehand to obtain the size of the necessary work area. The ddmatrix_adjust_s subroutine is called to copy the matrix to the work area, and then the ddeigen_* subroutine is called to calculate the eigenvalues and eigenvectors.

9.2 Diagonalizing the Hilbert matrix (hilbert5_sample_dd)

Hilbert matrices are typical badly conditioned matrices that are often used in benchmark tests for numerical calculations. Hilbert matrix of order 5 is as follows.

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

A quadruple precision matrix $A(=Ah+Al)$ is given by substituting the elements of the Hilbert matrix as the results of quadruple precision divisions for the high word and low word. The cstab_get_optdim subroutine is called beforehand to obtain the size of necessary work area. The ddmatrix_adjust_s subroutine is called to copy the matrix to the work area, and then the ddeigen_s subroutine is called to calculate the eigenvalues and eigenvectors.

9.3 Execution of sample programs

Those sample programs have already been compiled during the make, and you can, for example, execute frank5_sample_dd by entering as follows.
(The following examples are for QPEigen_s. For the other libraries, replace _s with the appropriate suffixes)

```
cd sample
./frank5_sample_dd
```

If you want to compile this program yourself, perform, for example, the following.

```
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/libddblas.a ../../lib/libscalapack.a ¥
 ../../lib/liblapack.a ../../lib/libblas.a
```

The source code of the sample program "frank5_sample_dd.f" is as follows.

```
      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 Subroutine details

Table 1 shows a list of the routine modules in the QPEigen_s library.

Table1. Subroutines of QPEigen_s

Routine name	Function
ddeigen_s	Calculating the eigenvalues and eigenvectors of a real symmetric matrix by tridiagonalizing the matrix using the Householder method. Executes a series of actions: tridiagonalization, eigenvalue calculation, and eigenvector calculation.
ddeigen_trd	Tridiagonalizes a real symmetric matrix using HouseHolder tridiagonalization.
ddeigen_dc	Calculates the eigenvalues of an input matrix using the divide-and-conquer algorithm if the input matrix is a tridiagonal matrix obtained by HouseHolder tridiagonalization of a real symmetric matrix.
ddeigen_tbk	Calculates the eigenvectors of the tridiagonal matrix obtained by HouseHolder tridiagonalization of a real symmetric matrix.
ddmatrix_adjust_s	The DDEigen library uses two-dimensional cyclic distribution. This conversion routine is used to set up the data distribution of a matrix to support cyclic distribution. This routine must be called before using this library.
ddcommunication_s	Module used for MPI communication

Table 2 shows a list of the routine modules in the QPEigen_sx library.

Table2. Subroutines of QPEigen_sx

Routine name	Function
ddeigen_sx	Calculating the eigenvalues and eigenvectors of a real symmetric matrix by pentadiagonalizing the matrix using the NarrowBandReduction method. Executes a series of actions: pentadiagonalization, eigenvalue calculation, and eigenvector calculation.
ddeigen_prd	Pentadiagonalizes a real symmetric matrix using NarrowBandReduction pentadiagonalization.
ddeigen_dcx	Calculates the eigenvalues of an input matrix using the divide-and-conquer algorithm if the input matrix is a pentadiagonal matrix obtained by NarrowBandReduction pentadiagonalization of a real symmetric matrix.
ddeigen_pbk	Calculates the eigenvectors of the pentadiagonal matrix obtained by NarrowBandReduction pentadiagonalization of a real symmetric matrix.
ddmatrix_adjust_sx	The DDEigen library uses two-dimensional cyclic distribution. This conversion routine is used to set up the data distribution of a matrix to support cyclic distribution. This routine must be called before using this library.
ddcommunication_sx	Module used for MPI communication

Table 3 shows a list of the routine modules in the QPEigen_h library.

Table3. Subroutines of QPEigen_h

Routine name	Function
ddeigen_h	Calculating the eigenvalues and eigenvectors of a Hermitian matrix.
ddeigen_hrd	Diagonalizes a Hermitian matrix.
ddeigen_dch	Calculates the eigenvalues of an input matrix using the divide-and-conquer algorithm.
ddeigen_hbk	Calculates the eigenvectors of the matrix.
ddmatrix_adjust_h	The DDEigen library uses two-dimensional cyclic distribution. This conversion routine is used to set up the data distribution of a matrix to support cyclic distribution. This routine must be called before using this library.This routine is used for real data.
ddmatrix_adjust_hi	This routine is used for imaginary data.
ddcommunication_h	Module used for MPI communication

Routine name	ddeigen_s			
Function	Calculating the eigenvalues and eigenvectors of a real symmetric matrix by tridiagonalizing the matrix using the Householder method. Executes a series of actions: tridiagonalization, eigenvalue calculation, and eigenvector calculation.			
Specification	ddeigen_s(n, ah, al, lda, wh, wl, zh, zl, ldz, m, ifl)			
Arguments	Descriptions		i/o	Remarks
n	Dimension of a matrix	integer	i	
ah	High word of quadruple precision real matrix	double	i	
al	Low word of quadruple precision real matrix	double	i	
lda	Size of array A	integer	i	
wh	High word of eigenvalues	double	o	
wl	Low word of eigenvalues	double	o	
zh	High word of eigenvectors	double	o	
zl	Low word of eigenvectors	double	o	
ldz	Size of array Z	integer	i	
m	Blocking factor	integer	i	about 32 to 128
ifl	Option for output eigenvectors	integer	i	0 or 1

Routine name	ddeigen_trd			
Function	Tridiagonalizes a real symmetric matrix using HouseHolder tridiagonalization			
Specification	ddeigen_trd(n, ah, al, lda, dh, dl, eh, el, m)			
Arguments	Descriptions		i/o	Remarks
n	Dimension of a matrix	integer	i	
ah	High word of quadruple precision real matrix	double	i	
al	Low word of quadruple precision real matrix	double	i	
lda	Size of array A	integer	i	
dh	High word of diagonal elements	double	o	
dl	Low word of diagonal elements	double	o	
eh	High word of subdiagonal elements	double	o	
el	Low word of subdiagonal elements	double	o	
m	Blocking factor	integer	i	

Routine name	ddeigen_dc			
Function	Calculates the eigenvalues of an input matrix using the divide-and-conquer algorithm if the input matrix is a tridiagonal matrix obtained by HouseHolder tridiagonalization of a real symmetric matrix.			
Specification	ddeigen_dc(n, wh, wl, eh, el, zh, zl, ldz, info)			
Arguments	Descriptions		i/o	Remarks
n	Dimension of a matrix	integer	i	
wh	High word of eigenvalues	double	o	
wl	Low word of eigenvalues	double	o	
eh	High word of subdiagonal elements	double	i	

el	Low word of subdiagonal elements	double	i	
zh	High word of eigenvectors	double	o	
zl	Low word of eigenvectors	double	o	
ldz	Size of array Z	integer	i	
info	Error number	integer	o	

Routine name	ddeigen_tbk			
Function	Calculates the eigenvectors of the tridiagonal matrix obtained by HouseHolder tridiagonalization of a real symmetric matrix.			
Specification	ddeigen_tbk(n, ah, al, lda, zh, zl, ldz, eh, el, m)			
Arguments	Descriptions		i/o	Remarks
n	Dimension of a matrix	integer	i	
ah	High word of the matrix	double	i	
al	Low word of the matrix	double	i	
lda	Size of array A	integer	i	
zh	High word of eigenvectors	double	o	
zl	Low word of eigenvectors	double	o	
ldz	Size of array Z	integer	i	
eh	High word of subdiagonal components	double	i	
el	Low word of subdiagonal components	double	i	
m	Blocking factor	integer	i	

Routine name	ddmatrix_adjust_s			
Function	The DDEigen library uses two-dimensional cyclic distribution. This conversion routine is used to set up the data distribution of a matrix to support cyclic distribution. This routine must be called before using this library.			
Specification	ddmatrix_adjust_s(n, ah, al, bh, bl, nm)			
Arguments	Descriptions		i/o	Remarks
n	Dimension of a matrix	integer	i	
ah	High word of original matrix	double	i	
al	Low word of original matrix	double	i	
bh	High word of distributed data	double	o	
bl	Low word of distributed data	double	o	
nm	Size of array b	integer	i	

Routine name	ddeigen_sx			
Function	Calculating the eigenvalues and eigenvectors of a real symmetric matrix by pentadiagonalizing the matrix using the NarrowBandReduction method. Executes a series of actions: pentadiagonalization, eigenvalue calculation, and eigenvector calculation.			
Specification	ddeigen_sx(n, ah, al, nma, wh, wl, zh, zl, dh, dl, eh, el, nme, m0, ifl)			
Arguments	Descriptions		i/o	Remarks
n	Dimension of a matrix	integer	i	
ah	High word of quadruple precision real matrix	double	i	
al	Low word of quadruple precision real matrix	double	i	
nma	Size of array A	integer	i	
wh	High word of eigenvalues	double	o	
wl	Low word of eigenvalues	double	o	
zh	High word of eigenvectors	double	o	
zl	Low word of eigenvectors	double	o	
dh	High word of array to store a main diagonal elements	double	o	
dl	Low word of array to store a main diagonal elements	double	o	
eh	High word of array to store a sub diagonal elements	double	o	
el	Low word of array to store a sub diagonal elements	double	o	
nme	Size of array e	integer	i	
m0	Blocking factor	integer	i	about 32 to 128
ifl	Option for output eigenvectors	integer	i	0 or 1

Routine name	ddeigen_prd			
Function	Pentadiagonalizes a real symmetric matrix using NarrowBandReduction pentadiagonalization			
Specification	ddeigen_prd(n, ah, al, nma, dh, dl, eh, el, nme, m0)			
Arguments	Descriptions		i/o	Remarks
n	Dimension of a matrix	integer	i	
ah	High word of quadruple precision real matrix	double	i	
al	Low word of quadruple precision real matrix	double	i	
nma	Size of array A	integer	i	
dh	High word of diagonal elements	double	o	
dl	Low word of diagonal elements	double	o	
eh	High word of subdiagonal elements	double	o	
el	Low word of subdiagonal elements	double	o	
nme	Size of array e	integer	i	
m0	Blocking factor	integer	i	about 32 to 128

Routine name	ddeigen_dcx			
Function	Calculates the eigenvalues of an input matrix using the divide-and-conquer algorithm if the input matrix is a pentadiagonal matrix obtained by NarrowBandReduction pentadiagonalization of a real symmetric matrix.			
Specification	ddeigen_dcx(n, wh, wl, eh, el, nme, zh, zl, nmz)			
Arguments	Descriptions		i/o	Remarks
n	Dimension of a matrix	integer	i	
wh	High word of eigenvalues	double	i	
wl	Low word of eigenvalues	double	i	
eh	High word of sub diagonal elements	double	i	
el	Low word of sub diagonal elements	double	i	
nme	Size of array e	integer	i	
zh	High word of eigenvectors	double	o	
zl	Low word of eigenvectors	double	o	
nmz	Size of array Z	integer	i	

Routine name	ddeigen_pbk			
Function	Calculates the eigenvectors of the pentadiagonal matrix obtained by NarrowBandReduction pentadiagonalization of a real symmetric matrix.			
Specification	ddeigen_pbk(n, ah, al, nma, zh, zl, nmz, eh, el, m0)			
Arguments	Descriptions		i/o	Remarks
n	Dimension of a matrix	integer	i	
ah	High word of quadruple precision real matrix	double	i	
al	Low word of quadruple precision real matrix	double	i	
nma	Size of array A	integer	i	
zh	High word of eigenvectors	double	o	
zl	Low word of eigenvectors	double	o	
nmz	Size of array Z	integer	i	
eh	High word of subdiagonal components	double	i	
el	Low word of subdiagonal components	double	i	
m0	Blocking factor	integer	i	about 32 to 128

Routine name	ddmatrix_adjust_sx			
Function	The DDEigen library uses two-dimensional cyclic distribution. This conversion routine is used to set up the data distribution of a matrix to support cyclic distribution. This routine must be called before using this library.			
Specification	ddmatrix_adjust_sx(n, ah, al, bh, bl, nm)			
Arguments	Descriptions		i/o	Remarks
n	Dimension of a matrix	integer	i	
ah	High word of original matrix	double	i	
al	Low word of original matrix	double	i	
bh	High word of distributed data	double	o	
bl	Low word of distributed data	double	o	
nm	Size of array b	integer	i	

Routine name	ddeigen_h			
Function	Calculating the eigenvalues and eigenvectors of a Hermitian matrix.			
Specification	ddeigen_h(n, arh, arl, aih, ail, wh, wl, lda, m0, ifl)			
Arguments	Descriptions		i/o	Remarks
n	Dimension of a matrix	integer	i	
arh	High word of the real part of the matrix (on exit) High word of the real part of eigenvectors	double	i/o	
arl	High word of the real part of the matrix (on exit) Low word of the real part of eigenvectors	double	i/o	
aih	Low word of the imaginary part of the matrix (on exit) High word of the imaginary part of eigenvectors	double	i/o	
ail	Low word of the imaginary part of the matrix (on exit) Low word of the imaginary part of eigenvectors	double	i/o	
wh	High word of eigenvalues	double	o	
wl	Low word of eigenvalues	double	o	
lda	Size of array A	integer	i	
m0	Blocking factor	integer	i	
ifl	Option for output eigenvectors	integer	i	0 or 1

Routine name	ddeigen_hrd			
Function	Calculating the eigenvalues and eigenvectors of a Hermitian matrix.			
Specification	ddeigen_hrd(n, arh, arl, aih, ail, lda, dh, dl, eh, el, e2h, e2l, tauh, taul, m, zrh, zrl, zih, zil)			
Arguments	Descriptions		i/o	Remarks
n	Dimension of a matrix	integer	i	
arh	High word of the real part of the matrix	double	i	
arl	High word of the real part of the matrix	double	i	
aih	Low word of the imaginary part of the matrix	double	i	
ail	Low word of the imaginary part of the matrix	double	i	
lda	Size of array A	integer	i	
dh	High word of diagonal elements	double	o	
dl	Low word of diagonal elements	double	o	
eh	High word of subdiagonal elements	double	o	

el	Low word of subdiagonal elements	double	o	
e2h	Work area	double	o	
e2l	Work area	double	o	
tauh	Work area	double	o	
taul	Work area	double	o	
m	Blocking factor	integer	i	about 128
zrh	High word of real part of the eigenvectors	double	o	
zrl	Low word of real part of the eigenvectors	double	o	
zih	High word of imaginary part of the eigenvectors	double	o	
zil	Low word of imaginary part of the eigenvectors	double	o	

Routine name	ddeigen_dch			
Function	**Calculates the eigenvalues of an input matrix using the divide-and-conquer algorithm.**			
Specification	ddeigen_dch(n, dh, dl, eh, el, zh, zl, ldz, info)			
Arguments	Descriptions		i/o	Remarks
n	Dimension of a matrix	integer	i	
dh	High word of eigenvalues	double	o	
dl	Low word of eigenvalues	double	o	
eh	Work area	double	o	
el	Work area	double	o	
zh	High word of real part of the eigenvectors	double	o	
zl	Low word of real part of the eigenvectors	double	o	
ldz	Size of array Z	integer	i	
info	Error number	integer	o	

Routine name	ddeigen_hbk			
Function	**Calculates the eigenvectors of the matrix.**			
Specification	ddeigen_hbk(n, arh, arl, aih, ail, lda, zrh, zrl, zih, zil, ldz, tauh, taul, ltau, n)			
Arguments	Descriptions		i/o	Remarks
arh	High word of the real part of the matrix	double	i	
arl	High word of the real part of the matrix	double	i	
aih	Low word of the imaginary part of the matrix	double	i	
ail	Low word of the imaginary part of the matrix	double	i	
lda	Size of array A	integer	i	
zrh	High word of the real part of the matrix	double	i	
zrl	High word of the real part of the matrix	double	i	

zih	Low word of the imaginary part of the matrix	double	i	
zil	Low word of the imaginary part of the matrix	double	i	
ldz	Size of array z	integer	i	
tauh	Work area	double	o	
taul	Work area	double	o	
ldtau	Work area	Integer	o	
n	Dimension of a matrix	integer	i	

Routine name	ddmatrix_adjust_h, ddmatrix_adjust_hi			
Function	The DDEigen library uses two-dimensional cyclic distribution. This conversion routine is used to set up the data distribution of a matrix to support cyclic distribution. This routine must be called before using this library.This routine is used for real data. Ddmatrix_adjust_hi is for imaginary data.			
Specification	ddmatrix_adjust_h(n, ah, al, bh, bl, nm) ddmatrix_adjust_hi(n, ah, al, bh, bl, nm)			
Arguments	Descriptions		i/o	Remarks
n	Dimension of a matrix	integer	i	
ah	High word of original matrix	double	i	
al	Low word of original matrix	double	i	
bh	High word of distributed data	double	o	
bl	Low word of distributed data	double	o	
nm	Size of array b	integer	i	