FUJITSU

FUJITSU Software
Technical Computing Suite V4.0L20

Development Studio
BLAS LAPACK ScaLAPACK
User's Guide

J2UL-2489-02ENZ0(00)
March 2020

Preface

The products are the implementations of the public domain BLAS (Basic Linear Algebra Subprograms), LAPACK (Linear Algebra PACKage) and ScaLAPACK (Scalable LAPACK) , which have been developed by groups of people such as Prof. Jack Dongarra, University of Tennessee, USA and all published on the WWW (URL: http://www.netlib.org/). This product includes not only the sequential versions of BLAS and LAPACK but also the thread-parallel versions which reduce turnaround time by a parallel algorithms.

The structure of the manual is as follows.

1 Overview

The products are outlined with emphasis on the differences from the public domain versions.

2 Documentation

The manual is not intended to give calling sequences of individual subprograms. Instead, the reader is requested to refer to a set of documentation available to the general public. The section describes where and how to access it.

3 Example program using BLAS and LAPACK

This section shows an example code that uses subroutines from BLAS and LAPACK versions. The example is simple enough to understand and intended for use in order to describe principles of calling thread-safe subroutines from an OpenMP Fortran program.

Appendix A Routines List

The entire list of subprograms provided is given.

Appendix B License

The manual describes the use of free software license by the products.

For general usage of BLAS, LAPACK and ScaLAPACK, please see the documentation mentioned in "2. Documentation".

In addition to this manual, the following manuals are references that apply to this manual:

Fortran User's Guide

MPI User's Guide

For a detailed specification of OpenMP, please refer the specification in http://www.openmp.org/.

Export Controls

Exportation/release of this document may require necessary procedures in accordance with the regulations of your resident country and/or US export control laws.

Date of Publication and Version

Version	Manual code
March 2020, 2nd Version	J2UL-2489-02ENZ0(00)
January 2020, 1st Version	J2UL-2489-01ENZ0(00)

Update History

Changes	Location	Version
The matrix copy and transpose routines was added.	1.2, A.4.2, A.4.3	Second Version
Changed the look according to product upgrades.	-	Second Version

Acknowledgement

BLAS, LAPACK and ScaLAPACK are collaborative effort involving several institution and it is distributed on Netlib.

Contents

List of Figures

List of Tables

1. Overview

BLAS, LAPACK and ScaLAPACK in the product are Fujitsu implementations of the netlib packages of BLAS (Basic Linear Algebra Subprograms), LAPACK (Linear Algebra PACKage) and ScaLAPACK (SCAlable LAPACK). The routines of BLAS and LAPACK can be called not only from sequential Fortran programs but also from thread-parallel programs written with OpenMP Fortran API. Furthermore, thread-parallel versions of BLAS and LAPACK are provided.

The routines of ScaLAPACK can be called from user programs written in Fortran with the CALL statement.

1.1 Differences between the sequential and thread-parallel versions

a) Sequential versions

The sequential BLAS and LAPACK routines can be called not only from sequential Fortran programs but also from parallel regions of OpenMP Fortran programs because BLAS/LAPACK routines in the product are thread-safe. When calling from parallel regions, the user can give different problem data to a single routine at the same time, so that the routine can deal with the problems simultaneously using multiple threads. For instance, when a matrix-matrix multiplication routine is called that way, one thread takes one problem of multiplication.

In the conventional sequential computation, the user needs to make subsequent calls to a subroutine by changing set of data repeatedly. With the BLAS thread-safe library, however, the user can give several sets of data to a subroutine at a time using multiple threads, where one thread takes care of one set of data and all the threads run concurrently. This way, the user can expect a parallel execution with the number of parallelism equal to the number of threads. Of course, multiple CPUs or cores have to be available for reduction of elapsed time. The user will see the example code in the section of “3 Example program using BLAS and LAPACK”

b) Thread-parallel versions

A routine from thread-parallel versions solves a single (rather big) problem by a parallel algorithm using multiple threads. As a result, the turnaround time can be reduced. For instance, when the user gives a problem to the thread-parallel routine for matrix-matrix multiplication, the routine creates multiple threads inside, distributes pieces of computation to the threads, and allows them to run on multiple threads. Creation of threads, work-sharing, synchronization, and termination of threads are controlled by OpenMP Fortran specifications.

Routine names and calling sequences of the thread-parallel routines are unchanged from those of the sequential routines, so the user does not need to modify any part of calls to BLAS/LAPACK. Just switch to the link library for the thread-parallel versions.

It cannot be possible to mix the sequential routines and thread-parallel routines in a single application.

In addition to LAPACK parallelized by OpenMP , PLASMA, which is parallelized by pthread, is provided. The routine names of PLASMA are different from LAPACK and both PLASMA and LAPACK routines are called in a single program.

1.2 BLAS

BLAS is a library for vector and matrix operations. This product is based on BLAS (1998-07-03 version) provided on Netlib. BLAS includes 81 functions. The total number of routines for all precision types amounts to approximately 660.

BLAS provides the following routines.

Level 1 BLAS	:	Vector operations
Level 2 BLAS	:	Matrix and vector operations
Level 3 BLAS	:	Matrix and matrix operations
Sparse-BLAS	:	Sparse vector operations
XBLAS Version 1.0.248	:	Extra Precise BLAS
CBLAS 2/23/03 version	:	C interface of BLAS

The routines in this BLAS library have exactly the same subroutine names and calling parameters as those of netlib baseline version.

Subroutines parallelized

All of the Level 3 BLAS and frequently used routines of the Level 2 BLAS have been parallelized. The rest of routines are just sequential even though they are contained in the thread-parallel library. For complete list of routines that are parallelized, see "A.1 BLAS" later in this manual.

There are extended functions in addition to standard BLAS routines in this product. See "A.4 BLAS-like extensions" for detail.

- Half precision (REAL*2) BLAS routines
- Sequential BLAS routines in the thread parallel BLAS library
- Matrix copy and transpose routines

1.3 LAPACK

LAPACK is a library of linear algebra routines. This product is based on LAPACK version 3.5.0 provided on Netlib. LAPACK includes approximately 400 functions. The total number of routines for all precision types amounts to approximately 1700.

PLASMA is also provided. This product is based on PLASMA version 2.7.1.

LAPACK provides the following routines.

- Linear equations
- Linear least squares problems
- Eigenvalue problems
- Singular value decomposition

LAPACK, a collection of subroutines for linear algebra is, like thread-safe BLAS, called not only from sequential Fortran programs but also a program written in OpenMP Fortran in the environment of SMP. The purpose of using it is to have a subroutine concurrently solve different problems that are independent from each other and so reduce the turnaround time to solve all the problems.

LAPACK contains driver routines, computational routines and auxiliary routines. Driver routines are those which deal with general linear algebraic problems such as a system of linear equations, while computational routines serve to work as components of driver routines such as LU decomposition of matrices. Auxiliary routines perform certain subtask or common low-level computation.

Subroutines parallelized

Functions listed below have been parallelized by OpenMP. For complete list of routines, see "A.2 LAPACK" later in this manual. Large part of the other routines has calls to BLAS that are parallelized, so it can be said most of LAPACK are more or less parallelized.

- Linear equations for a general matrix, a symmetric/ Hermitian positive definite matrix and a symmetric/Hermitian matrix. (simple driver and expert driver)

- Symmetric/Hermitian eigenvalue problem using the divide and conquer algorithm.
- Generalized Symmetric/Hermitian eigenvalue problem using the divide and conquer algorithm.
- Linear least squares problems.
- Singular value decomposition.

1.4 ScaLAPACK

ScaLAPACK is a library of high performance linear algebra routines for distributed memory message-passing computers. It can solve systems of linear equations, linear least squares problems, eigenvalue problems and singular value problems.

ScaLAPACK is built on two layers: PBLAS, BLACS. The PBLAS includes subroutines for common linear algebra computation for parallel processors. The BLACS is a communication library for linear algebra.

ScaLAPACK, PBLAS and BLACS provide the following routines.

ScaLAPACK	:	Linear equations Linear least squares problems Eigenvalue problems Singular value decomposition
PBLAS	:	Level 1 PBLAS Vector operations Level 2 PBLAS Matrix and vector operations Level 3 PBLAS Matrix and matrix operations
BLACS	:	Send and receive a matrix or submatrix from one processor to another using MPI.

This product is based on the following versions of software on Netlib.

ScaLAPACK	:	Version 2.0.2
PBLAS	:	Source programs as a part of ScaLAPACK Version 2.0.2
BLACS	:	Version 1.1

ScaLAPACK layer contains driver routines, computational routines and auxiliary routines.

ScaLAPACK includes approximately 200 functions. The total number of routines for all precision types amounts to approximately 700. Each routine can be called from user programs written in Fortran with the CALL statement.

PBLAS layer is a MPI parallel version of BLAS, which provides a subset of BLAS routines. ScaLAPACK layer is a MPI parallel version of LAPACK, which provides a subset of LAPACK routines.

ScaLAPACK provides flexible methods for partitioning the global matrix among the processors, but each routine has appropriate ways of partitioning for better performance. See Netlib Web page to get more information about the distribution parameters. When the user pays attention to these parameters, ScaLAPACK realizes the good scalability.

ScaLAPACK routines call BLAS and LAPACK routines. User can choose either sequential version or thread-parallel version of BLAS and LAPACK library. In a user program which is link-edited with BLAS and LAPACK thread-parallel version, ScaLAPACK routines are executed in parallel using MPI in each of which BLAS and LAPACK routines that are called from ScaLAPACK routines are executed in parallel by multiple threads. Generally, when some processors are assigned to thread-parallel calculation, the performance is better because the balance of calculation is better and the communication is smaller. But if the routine executes most of the calculation in ScaLAPACK layer and doesn't depend on BLAS and LAPACK routines, the performance maybe slower. So, it is needed to pay attention to the balance of processors assigned to processes and threads.

ScaLAPACK routines are not thread-safe. It means that ScaLAPACK routines should not be called from multiple threads simultaneously in the parallel region in OpenMP Fortran programs.

1.5 Structure of BLAS, LAPACK and ScaLAPACK

The Figure 1 below depicts call tree.

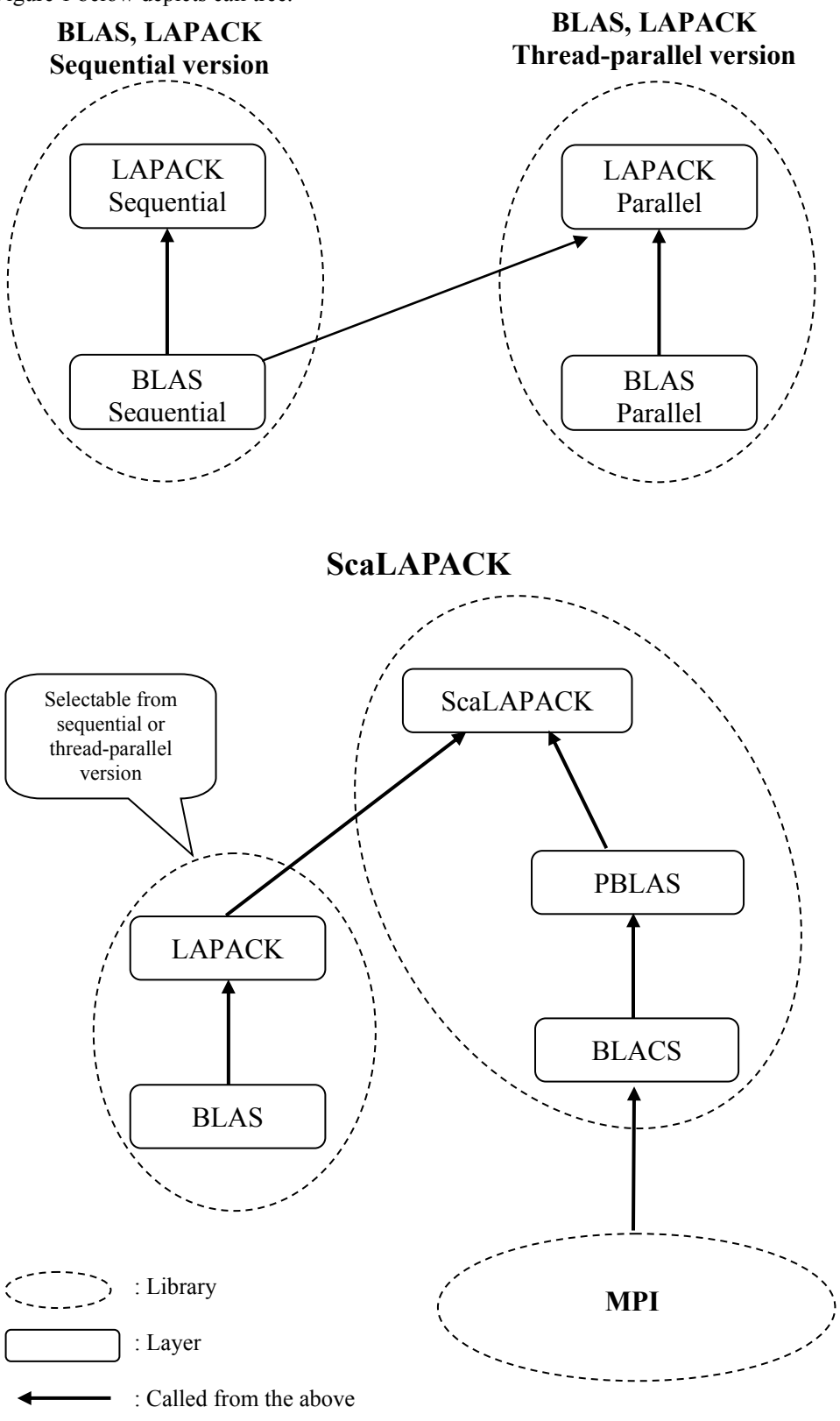

Figure 1 Structure of BLAS, LAPACK and ScaLAPACK

2. Documentation

The calling interfaces (routine name, parameter sequence) of subroutines provided in the products are unchanged from the public domain counterparts. For that reason, the manual does not include the calling specifications of individual subroutines. Instead, the user is requested to refer to the publicly available documentation to be listed below.

2.1 Users' Guide

Refer to the following manual for usage descriptions of the individual routines provided by this software:

- *LAPACK Users' Guide, Third Edition* (SIAM, 1999)

  The book describes in detail the usage of LAPACK, including calling specifications, purposes, parameter descriptions, performances, and accuracy of driver routines and computational routines. Also included is a quick reference of BLAS.

- *ScaLAPACK Users' Guide* (SIAM, 1997)

  The book describes in detail the usage of ScaLAPACK, including ways of partitioning and distributing matrices among processors, calling specifications, purposes, parameter descriptions, performances, and accuracy of driver and computational routines. Also included is a list of error messages ScaLAPACK might issues. Finally, a quick references of PBLAS and BLACS is given.

  What is interesting, a CD-ROM comes with the book, packing all the relevant documents and source programs.

2.2 On-line Documentation

Several documentation of BLAS, LAPACK, and ScaLAPACK are available online from the domain http://www.netlib.org/. The following names are all as of February, 2019.

- BLAS
  http://www.netlib.org/blas/
- XBLAS
  http://www.netlib.org/xblas/
- CBLAS
  http://www.netlib.org/blas/blast-forum/
- LAPACK
  https://bitbucket.org/icl/plasma
- LAPACKE(C interface of LAPACK)
  http://www.netlib.org/lapack/lapacke.html
- PLASMA
  https://bitbucket.org/icl/plasma
- ScaLAPACK
  http://www.netlib.org/scalapack/

3. Example program using BLAS and LAPACK

This section describes how to call subroutines of BLAS and LAPACK from an OpenMP Fortran program by using a simple example code.

3.1 Problem

Let's consider a system of linear equations

$Ax = b$

, where A is a real dense matrix of order n, b the right hand side vector of order n, and x the solution vector. An interesting case is that we have multiple right hand side vectors for each of which we need the solution. This problem can be written in a matrix form,

$AX = B$

, where each column vector of B , denoted by b_i (i = 1,2,...,m) ,stands for each right hand side vector, and each column of X , denoted by x_i (i = 1,2,...,m) , the solution vector corresponding to b_i

3.2 Sequential version

This section describes how the user can take advantage of thread-safety of sequential versions of BLAS/LAPACK. First, let's take a look at an example program of conventional sequential codes

a) Calls from sequential programs

Let's assume the user wants to solve linear equations of order 200 with 80 of right hand side vectors. There is the driver subroutine DGESV from LAPACK which does all computations to get the solutions. For the purpose of explanation, however, the following example chooses to use the computational routines; DGETRF for LU decomposition and DGETRS for getting solutions using the LU factors. In the program, subroutine inita(..) is to set up the array "a" and initb(..) the 80 right hand side vectors.

```
      implicit real*8 (a-h,o-z)
      parameter(maxn=200,m=80,k=maxn+1)
      real*8 a(k,maxn),b(k,m)
      integer ip(maxn)
C ============================================================
C Define the matrix
C ============================================================
      n=maxn
      call inita(a,k,n)
      call initb(b,k,n,m)
C ============================================================
C LU decomposition
C ============================================================
      call dgetrf(n,n,a,k,ip,info)
      if(info.ne. 0) then
        print *,'The given problem seems to be not normal'
        stop
      endif
C ============================================================
C Solution
C ============================================================
```

```
        call dgetrs('N',n,m,a,k,ip,b,k,info)
                :
```

b) Calls from an OpenMP Fortran program

In the above example, subroutine DGETRS gets 80 of the solution vectors corresponding to 80 of the right hand side vectors, where each solution vector can be obtained independently and so in parallel with the rest of solutions. Let's assume that 80 solution vectors are divided into 20 subsets, each having 4 solutions, and have one subset handled by one thread. Each thread calls DGETRS with a subset of 4 right hand side vectors. The following example shows how this is done by using OpenMP Fortran directives.

```
      implicit real*8 (a-h,o-z)
      parameter(maxn=200,m=80,mblk=4,k=maxn)
      real*8 a(k,maxn),b(k,m)
      integer ip(maxn)
C ============================================================
C Define the matrix
C ============================================================
      n=maxn
      call inita(a,k,n)
      call initb(b,k,n,m)
C ============================================================
C LU decomposition
C ============================================================
      call dgetrf(n,n,a,k,ip,info)
      if(info.ne. 0) then
        print *,'The given problem seems to be not normal'
        stop
      endif
C ============================================================
C Solution
C ============================================================
!$OMP PARALLEL DO PRIVATE(mb,info)
      do i=1,m,mblk
        mb=min(mblk,m-i+1)
        call dgetrs('N',n,mb,a,k,ip,b(1,i),k,info)
      end do
!$OMP END PARALLEL DO
              :
```

Explanations

1. The !$OMP PARALLEL and !$OMP END PARALLEL directive pair define a parallel region where the code block between the directive pair executes in parallel by multiple threads. The !$OMP PARALLEL creates a team of multiple threads and the clause PRIVATE(...) specifies that variables or arrays in the parentheses are allocated memory copy per each thread. Variables or arrays that do not appear in the PRIVATE clause have only one copy and shared by all threads in the parallel region.
2. The directive !$OMP DO specifies that DO construct right below the directive can be executed in parallel, with respect to the DO index, by multiple threads. In other words, computation for each value of DO index is done by a thread asynchronously with the rest of threads.
3. What is one more important characteristic is that the above code can be compiled without option `-Kopenmp` features, link-edited with BLAS and LAPACK libraries, then can work correctly producing the same results, while execution takes longer. This is because the OpenMP directives are treated as Fortran comment statements and executable Fortran statements have nothing different from regular Fortran constructs.

3.3 Thread-parallel version

Let's consider now how a thread-parallel subroutine of BLAS/LAPACK can be utilized. First, let's take a look at an example program of conventional sequential codes that calls a thread-parallel

subroutine. Then we will move on to an example where a thread-parallel routine is called from a parallelized program by OpenMP Fortran.

a) Calls from sequential programs

Let's assume the user wants to solve linear equations of order 3000 with 800 of right hand side vectors. This time, we choose to use subroutine DGESV from LAPACK, which does all computations to get the solutions from LU decomposition through solution with the LU factors.

Compile the following sequential program and link-edit with the thread-parallel library in stead of the sequential library. What will happen then is that inside DGESV, multiple threads are created and computation is shared by the multiple threads, i.e. multiple CPUs or cores. (This situation happens even when the thread-parallel routine is called from sequential portions of OpenMP Fortran programs.)

```
      implicit real*8 (a-h,o-z)
      parameter(maxn=3000,m=800,k=maxn+1)
      real*8 a(k,maxn),b(k,m)
      integer ip(maxn)
C ============================================================
C Define the matrix
C ============================================================
      n=maxn
      call inita(a,k,n)
      call initb(b,k,n,m)
C ============================================================
C LU decomposition and Solution
C ============================================================
      call dgesv(n,m,a,k,ip,b,k,info)
      if(info.ne. 0) then
        print *,'The given problem seems to be not normal'
        stop
      endif
            :
```

b) Calls from an OpenMP Fortran program

A thread-parallel subroutine can be called from inside parallelized portions of OpenMP Fortran. Let's assume that there are two systems of linear equations which are independent from each other and the user wishes to solve each of systems by calling the thread-parallel version of subroutine DGESV.

In the following example, each equation is solved by two threads. The user is requested to set the environment variable OMP_NUM_THREADS to 2. Then, two sets of equations will be solved on a total of 4 cores, where each uses 2 cores.

```
      implicit real*8 (a-h,o-z)
      parameter(maxn=3000,m=800,k=maxn,np=2)
      real*8 a(k,maxn,np),b(k,m,np)
      integer nsize(np)
      data nsize/2800,3000/
      integer ip(maxn,np)
!$OMP PARALLEL DO DEFAULT(SHARED) PRIVATE(n,info)
      do ii=1,np
C ============================================================
C Define the matrix
C ============================================================
        n=nsize(ii)
        call inita(a(1,1,ii),k,n)
        call initb(b(1,1,ii),k,n,m)
C ============================================================
C LU decomposition and Solution
C ============================================================
        call dgesv(n,m,a(1,1,ii),k,ip(1,ii),b(1,1,ii),k,info)
        if(info.ne. 0) then
          print *,'The given problem seems to be not normal'
          stop
        endif
      end do
!$OMP END PARALLEL DO
```

:

Explanation

1. There is one thing that should be taken care of by the user to run the above program. That is, the user needs to allow parallel regions to be nested. This can be done by switching on the environment variable OMP_NESTED, or by using the run time library routine OMP_SET_NESTED.

   The above program could run on less than 4 cores, apart from performance.

Appendix A Routines List

In order to help the user make sure the routines interested are provided in the product, the entire lists of routines are provided here. The user is asked to check with the following lists whenever he/she feels uncertain about availability. Note, however, that the coverage of routines here does not always keep up with the latest versions from Netlib. The lists below are intended to be used to check the difference, if any, from the Netlib.

A.1 BLAS

The routines that are supplied with BLAS are listed in Table A.1 to Table A.7. The slave routines included in this software are listed in Table A.8

The mark # means that it takes either of:

S : REAL
D : DOUBLE PRECISION
C : COMPLEX
Z : COMPLEX*16

A combination of precisions means to use more than one precision. For example, “SC” of SCNRM2 is a function returning real and complex entries.

The marks in the column of Parallel shows following means:

O : Parallelized in the thread-parallel version
– : Not parallelized (same as sequential routine)

Table A.1 Level 1 BLAS routines

Fortran BLAS Routine name	CBLAS Routine name	Supported precision (small letter for CBLAS)	Parallel
#ROTG	cblas_#rotg	S, D	–
#ROTMG	cblas_#rotmg	S, D	–
#ROT	cblas_#rot	S, D	O
#ROTM	cblas_#rotm	S, D	O
#SWAP	cblas_#swap	S, D, C, Z	O
#SCAL	cblas_#scal	S, D, C, Z, CS, ZD	O
#COPY	cblas_#copy	S, D, C, Z	O
#AXPY	cblas_#axpy	S, D, C, Z	O
#DOT	cblas_#dot	S, D, DS	O
#DOTU	cblas_#dotu_sub	C, Z	O
#DOTC	cblas_#dotc_sub	C, Z	O
##DOT	cblas_##dot	SDS	O
#NRM2	cblas_#nrm2	S, D, SC, DZ	O
#ASUM	cblas_#asum	S, D, SC, DZ	O
I#AMAX	cblas_i#amax	S, D, C, Z	O

Table A.2 Level 2 BLAS routines

Fortran BLAS Routine name	CBLAS Routine name	Supported precision (small letter for CBLAS)	Parallel
#GEMV	cblas_#gemv	S, D, C, Z	O
#GBMV	cblas_#gbmv	S, D, C, Z	–
#HEMV	cblas_#hemv	C, Z	O
#HBMV	cblas_#hbmv	C, Z	–
#HPMV	cblas_#hpmv	C, Z	O
#SYMV	cblas_#symv	S, D	O
#SBMV	cblas_#sbmv	S, D	–
#SPMV	cblas_#spmv	S, D	O
#TRMV	cblas_#trmv	S, D, C, Z	O
#TBMV	cblas_#tbmv	S, D, C, Z	–
#TPMV	cblas_#tpmv	S, D, C, Z	–
#TRSV	cblas_#trsv	S, D, C, Z	O
#TBSV	cblas_#tbsv	S, D, C, Z	–
#TPSV	cblas_#tpsv	S, D, C, Z	–
#GER	cblas_#ger	S, D	O
#GERU	cblas_#geru	C, Z	O
#GERC	cblas_#gerc	C, Z	O
#HER	cblas_#her	C, Z	O
#HPR	cblas_#hpr	C, Z	–
#HER2	cblas_#her2	C, Z	O
#HPR2	cblas_#hpr2	C, Z	–
#SYR	cblas_#syr	S, D	O
#SPR	cblas_#spr	S, D	–
#SYR2	cblas_#syr2	S, D	O
#SPR2	cblas_#spr2	S, D	–

Table A.3 Level 3 BLAS routines

Fortran BLAS Routine name	CBLAS Routine name	Supported precision (small letter for CBLAS)	Parallel
#GEMM	cblas_#gemm	S, D, C, Z	O
#SYMM	cblas_#symm	S, D, C, Z	O
#HEMM	cblas_#hemm	C, Z	O
#SYRK	cblas_#syrk	S, D, C, Z	O
#HERK	cblas_#herk	C, Z	O
#SYR2K	cblas_#syr2k	S, D, C, Z	O
#HER2K	cblas_#her2k	C, Z	O
#TRMM	cblas_#trmm	S, D, C, Z	O
#TRSM	cblas_#trsm	S, D, C, Z	O

Table A.4 Sparse BLAS(Fortran only)

Routine name	Supported precision	Parallel
#AXPYI	S, D, C, Z	–
#DOTI	S, D	–
#DOTCI	C, Z	–
#DOTUI	C, Z	–
#GTHR	S, D, C, Z	–
#GTHRZ	S, D, C, Z	–
#SCTR	S, D, C, Z	–
#ROTI	S, D	–

Table A.5 Level 1 XBLAS routines (Fortran/C)

Routine name	Precision (#)	Precision ($)	Parallel
BLAS_#dot_$	s	x	–
	d	d_s, d_s_x, s_d, s_d_x, s_s, s_s_x, x	–
	c	c_s, c_s_x, s_c, s_c_x, s_s, s_s_x, x	–
	z	c_c, c_c_x, c_z, c_z_x, d_d, d_d_x, d_z, d_z_x, x, z_c, z_c_x, z_d, z_d_x	–
BLAS_#sum_$	s	x	–
	d	x	–
	c	x	–
	z	x	–
BLAS_#axpby_$	s	x	–
	d	s, s_x, x	–
	c	s, s_x, x	–
	z	c, c_x, d, d_x, x	–
BLAS_#waxpby_$	s	x	–
	d	d_s, d_s_x, s_d, s_d_x, s_s, s_s_x, x	–
	c	c_s, c_s_x, s_c, s_c_x, s_s, s_s_x, x	–
	z	c_c, c_c_x, c_z, c_z_x, d_d, d_d_x, d_z, d_z_x, x, z_c, z_c_x, z_d, z_d_x	–

Table A.6 Level 2 XBLAS routines (Fortran/C)

Routine name	Precision (#)	Precision ($)	Parallel
BLAS_#gemv_$	s	x	–
	d	d_s, d_s_x, s_d, s_d_x, s_s, s_s_x, x	–
	c	c_s, c_s_x, s_c, s_c_x, s_s, s_s_x, x	–
	z	c_c, c_c_x, c_z, c_z_x, d_d, d_d_x, d_z, d_z_x, x, z_c, z_c_x, z_d, z_d_x	–
BLAS_#gemv2_$	s	x	–
	d	d_s, d_s_x, s_d, s_d_x, s_s, s_s_x, x	–
	c	c_s, c_s_x, s_c, s_c_x, s_s, s_s_x, x	–
	z	c_c, c_c_x, c_z, c_z_x, d_d, d_d_x, d_z, d_z_x, x, z_c, z_c_x, z_d, z_d_x	–
BLAS_#gbmv_$	s	x	–
	d	d_s, d_s_x, s_d, s_d_x, s_s, s_s_x, x	–

Table A.6 Level 2 XBLAS routines (Fortran/C) (continued)

Routine name	Precision (#)	Precision ($)	Parallel
BLAS_#gbmv_$	c	c_s, c_s_x, s_c, s_c_x, s_s, s_s_x, x	–
	z	c_c, c_c_x, c_z, c_z_x, d_d, d_d_x, d_z, d_z_x, x, z_c, z_c_x, z_d, z_d_x	–
BLAS_#gbmv2_$	s	x	–
	d	d_s, d_s_x, s_d, s_d_x, s_s, s_s_x, x	–
	c	c_s, c_s_x, s_c, s_c_x, s_s, s_s_x, x	–
	z	c_c, c_c_x, c_z, c_z_x, d_d, d_d_x, d_z, d_z_x, x, z_c, z_c_x, z_d, z_d_x	–
BLAS_#spmv_$	s	x	–
	d	d_s, d_s_x, s_d, s_d_x, s_s, s_s_x, x	–
	c	c_s, c_s_x, s_c, s_c_x, s_s, s_s_x, x	–
	z	c_c, c_c_x, c_z, c_z_x, d_d, d_d_x, d_z, d_z_x, x, z_c, z_c_x, z_d, z_d_x	–
BLAS_#hpmv_$	c	c_s, c_s_x, x	–
	z	c_c, c_c_x, c_z, c_z_x, x, z_c, z_c_x, z_d, z_d_x	–
BLAS_#symv_$	s	x	–
	d	d_s, d_s_x, s_d, s_d_x, s_s, s_s_x, x	–
	c	c_s, c_s_x, s_c, s_c_x, s_s, s_s_x, x	–
	z	c_c, c_c_x, c_z, c_z_x, d_d, d_d_x, d_z, d_z_x, x, z_c, z_c_x, z_d, z_d_x	–
BLAS_#symv2_$	s	x	–
	d	d_s, d_s_x, s_d, s_d_x, s_s, s_s_x, x	–
	c	c_s, c_s_x, s_c, s_c_x, s_s, s_s_x, x	–
	z	c_c, c_c_x, c_z, c_z_x, d_d, d_d_x, d_z, d_z_x, x, z_c, z_c_x, z_d, z_d_x	–
BLAS_#hemv_$	c	c_s, c_s_x, x	–
	z	c_c, c_c_x, c_z, c_z_x, x, z_c, z_c_x, z_d, z_d_x	–
BLAS_#hemv2_$	c	c_s, c_s_x, x	–
	z	c_c, c_c_x, c_z, c_z_x, x, z_c, z_c_x, z_d, z_d_x	–
BLAS_#sbmv_$	s	x	–
	d	d_s, d_s_x, s_d, s_d_x, s_s, s_s_x, x	–
	c	c_s, c_s_x, s_c, s_c_x, s_s, s_s_x, x	–
	z	c_c, c_c_x, c_z, c_z_x, d_d, d_d_x, d_z, d_z_x, x, z_c, z_c_x, z_d, z_d_x	–
BLAS_#hbmv_$	c	c_s, c_s_x, x	–
	z	c_c, c_c_x, c_z, c_z_x, x, z_c, z_c_x, z_d, z_d_x	–

Table A.6 Level 2 XBLAS routines (Fortran/C) (continued)

Routine name	Precision (#)	Precision ($)	Parallel
BLAS_#trsv_$	s	x	–
	d	s, s_x, x	–
	c	s, s_x, x	–
	z	c, c_x, d, d_x, x	–
BLAS_#tbsv_$	s	x	–
	d	s, s_x, x	–
	c	s, s_x, x	–
	z	c, c_x, d, d_x, x	–
BLAS_#trmv_$	s	x	–
	d	s, s_x, x	–
	c	s, s_x, x	–
	z	c, c_x, d, d_x, x	–
BLAS_#tpmv_$	s	x	–
	d	s, s_x, x	–
	c	s, s_x, x	–
	z	c, c_x, d, d_x, x	–
BLAS_#ge_sum_mv_$	s	x	–
	d	d_s, d_s_x, s_d, s_d_x, s_s, s_s_x, x	–
	c	c_s, c_s_x, s_c, s_c_x, s_s, s_s_x, x	–
	z	c_c, c_c_x, c_z, c_z_x, d_d, d_d_x, d_z, d_z_x, x, z_c, z_c_x, z_d, z_d_x	–

Table A.7 Level 3 XBLAS routines (Fortran/C)

Routine name	Precision (#)	Precision ($)	Parallel
BLAS_#gemm_$	s	x	–
	d	d_s, d_s_x, s_d, s_d_x, s_s, s_s_x, x	–
	c	c_s, c_s_x, s_c, s_c_x, s_s, s_s_x, x	–
	z	c_c, c_c_x, c_z, c_z_x, d_d, d_d_x, d_z, d_z_x, x, z_c, z_c_x, z_d, z_d_x	–
BLAS_#symm_$	s	x	–
	d	d_s, d_s_x, s_d, s_d_x, s_s, s_s_x, x	–
	c	c_s, c_s_x, s_c, s_c_x, s_s, s_s_x, x	–
	z	c_c, c_c_x, c_z, c_z_x, d_d, d_d_x, d_z, d_z_x, x, z_c, z_c_x, z_d, z_d_x	–
BLAS_#hemm_$	c	c_s, c_s_x, x	–
	z	c_c, c_c_x, c_z, c_z_x, x, z_c, z_c_x, z_d, z_d_x	–

Table A.8 Slave routines for BLAS

Routine name	Contents
DCABS1, LSAME, XERBLA	Included opened BLAS

A.2 LAPACK

The routines that are supplied with LAPACK are listed in Table A.9 to Table A.10.

The routine names in the following tables are the names of real and complex routines. The character indicating the supported precision is the first character of the routine name. For a double precision routine, replace the “S” of the real routine with “D”. For a double complex routine, replace the “C” of the complex routine with “Z”.

Auxiliary routine XERBLA and XERBLA_ARRAY is unique and do not depend on data type.

The marks in the column of Parallel shows following means:

O : Parallelized in the thread-parallel version
– : Not parallelized. Large part of the other routines have calls to BLAS that are parallelized, so it can be said most of LAPACK are more or less parallelized.

The symbol + shows new routines in version 3.4.1 of LAPACK on Netlib.

The symbol * shows new routines in version 3.5.0 of LAPACK on Netlib.

Table A.9 Driver and Computational routines of LAPACK

Real routine	Complex routine	Parallel
DSGESV	ZCGESV	—
DSPOSV	ZCPOSV	—
SBBCSD+	CBBCSD+	—
SBDSDC	—	O
SBDSQR	CBDSQR	—
SDISNA	—	—
SGBBRD	CGBBRD	—
SGBCON	CGBCON	—
SGBEQU	CGBEQU	—
SGBEQUB	CGBEQUB	—
SGBRFS	CGBRFS	—
SGBRFSX	CGBRFSX	—
SGBSV	CGBSV	—
SGBSVX	CGBSVX	—
SGBSVXX	CGBSVXX	—
SGBTRF	CGBTRF	—
SGBTRS	CGBTRS	—
SGEBAK	CGEBAK	—
SGEBAL	CGEBAL	—
SGEBRD	CGEBRD	O
SGECON	CGECON	—
SGEEQU	CGEEQU	O
SGEES	CGEES	—
SGEESX	CGEESX	—
SGEEQUB	CGEEQUB	—
SGEEV	CGEEV	—
SGEEVX	CGEEVX	—
SGEHRD	CGEHRD	O
SGEJSV	—	—
SGELQF	CGELQF	O
SGELS	CGELS	O
SGELSD	CGELSD	O
SGELSS	CGELSS	—

Real routine	Complex routine	Parallel
SGELSY	CGELSY	—
SGEMQRT+	CGEMQRT+	—
SGEQLF	CGEQLF	O
SGEQP3	CGEQP3	—
SGEQRF	CGEQRF	O
SGEQRFP	CGEQRFP	—
SGEQRT+	CGEQRT+	—
SGEQRT2+	CGEQRT2+	—
SGEQRT3+	CGEQRT3+	—
SGERFS	CGERFS	O
SGERFSX	CGERFSX	—
SGERQF	CGERQF	O
SGESDD	CGESDD	O
SGESV	CGESV	O
SGESVD	CGESVD	—
SGESVJ	—	—
SGESVX	CGESVX	O
SGESVXX	CGESVXX	—
SGETRF	CGETRF	O
SGETRI	CGETRI	—
SGETRS	CGETRS	—
SGGBAK	CGGBAK	—
SGGBAL	CGGBAL	—
SGGES	CGGES	—
SGGESX	CGGESX	—
SGGEV	CGGEV	—
SGGEVX	CGGEVX	—
SGGGLM	CGGGLM	O
SGGHRD	CGGHRD	—
SGGLSE	CGGLSE	O
SGGQRF	CGGQRF	O
SGGRQF	CGGRQF	O
SGGSVD	CGGSVD	—

Table A.9 Driver and Computational routines of LAPACK (continued)

Real routine	Complex routine	Parallel
SGGSVP	CGGSVP	—
SGTCON	CGTCON	—
SGTRFS	CGTRFS	—
SGTSV	CGTSV	—
SGTSVX	CGTSVX	—
SGTTRF	CGTTRF	—
SGTTRS	CGTTRS	—
—	CHEEQUB	—
—	CHERFSX	—
—	CHESVXX	—
SHGEQZ	CHGEQZ	—
SHSEIN	CHSEIN	—
SHSEQR	CHSEQR	—
SLARTGS^{+}	—	—
SLA_GERPVGRW^{+}	CLA_GERPVGRW^{+}	—
SOPGTR	CUPGTR	—
SOPMTR	CUPMTR	—
SORBDB^{+}	CUNBDB^{+}	—
SORBDB1 *	CUNBDB1 *	—
SORBDB2 *	CUNBDB2 *	—
SORBDB3 *	CUNBDB3 *	—
SORBDB4 *	CUNBDB4 *	—
SORBDB5 *	CUNBDB5 *	—
SORBDB6 *	CUNBDB6 *	—
SORCSD^{+}	CUNCSD^{+}	—
SORCSD2BY1 *	CUNCSD2BY1 *	—
SORGBR	CUNGBR	O
SORGHR	CUNGHR	—
SORGLQ	CUNGLQ	O
SORGQL	CUNGQL	O
SORGQR	CUNGQR	O
SORGRQ	CUNGRQ	O
SORGTR	CUNGTR	O
SORMBR	CUNMBR	O
SORMHR	CUNMHR	—
SORMLQ	CUNMLQ	O
SORMQL	CUNMQL	O
SORMQR	CUNMQR	O
SORMRQ	CUNMRQ	O
SORMRZ	CUNMRZ	—
SORMTR	CUNMTR	O
SPBCON	CPBCON	—
SPBEQU	CPBEQU	—
SPBRFS	CPBRFS	—
SPBSTF	CPBSTF	—

Real routine	Complex routine	Parallel
SPBSV	CPBSV	—
SPBSVX	CPBSVX	—
SPBTRF	CPBTRF	—
SPBTRS	CPBTRS	—
SPFTRF	CPFTRF	—
SPFTRI	CPFTRI	—
SPFTRS	CPFTRS	—
SPOCON	CPOCON	O
SPOEQU	CPOEQU	O
SPOEQUB	CPOEQUB	—
SPORFS	CPORFS	O
SPORFSX	CPORFSX	—
SPOSV	CPOSV	O
SPOSVX	CPOSVX	O
SPOSVXX	CPOSVXX	—
SPSTF2	CPSTF2	—
SPSTRF	CPSTRF	—
SPOTRF	CPOTRF	O
SPOTRI	CPOTRI	—
SPOTRS	CPOTRS	—
SPPCON	CPPCON	—
SPPEQU	CPPEQU	—
SPPRFS	CPPRFS	—
SPPSV	CPPSV	—
SPPSVX	CPPSVX	—
SPPTRF	CPPTRF	—
SPPTRI	CPPTRI	—
SPPTRS	CPPTRS	—
SPTCON	CPTCON	—
SPTEQR	CPTEQR	—
SPTRFS	CPTRFS	—
SPTSV	CPTSV	—
SPTSVX	CPTSVX	—
SPTTRF	CPTTRF	—
SPTTRS	CPTTRS	—
SSBEV	CHBEV	—
SSBEVD	CHBEVD	—
SSBEVX	CHBEVX	—
SSBGST	CHBGST	—
SSBGV	CHBGV	—
SSBGVD	CHBGVD	—
SSBGVX	CHBGVX	—
SSBTRD	CHBTRD	—
SSPCON	CSPCON	—
—	CHPCON	—

Table A.9 Driver and Computational routines of LAPACK (continued)

Real routine	Complex routine	Parallel	Real routine	Complex routine	Parallel
SSPEV	CHPEV	—	—	CHERFS	O
SSPEVD	CHPEVD	—	SSYSV	CSYSV	O
SSPEVX	CHPEVX	—	—	CHESV	O
SSPGST	CHPGST	—	SSYSV_ROOK *	CSYSV_ROOK *	O
SSPGV	CHPGV	—	—	CHESV_ROOK *	O
SSPGVD	CHPGVD	—	SSYSVX	CSYSVX	O
SSPGVX	CHPGVX	—	SSYSVXX	CSYSVXX	—
SSPRFS	CSPRFS	—	—	CHESVX	O
—	CHPRFS	—	SSYTRD	CHETRD	O
SSPSV	CSPSV	—	SSYTF2_ROOK *	CSYTF2_ROOK *	—
—	CHPSV	—	—	CHETF2_ROOK *	—
SSPSVX	CSPSVX	—	SSYTRF	CSYTRF	O
—	CHPSVX	—	—	CHETRF	O
SSPTRD	CHPTRD	—	SSYTRF_ROOK *	CSYTF2_ROOK *	O
SSPTRF	CSPTRF	—	—	CHETF2_ROOK *	O
—	CHPTRF	—	SSYTRI	CSYTRI	—
SSPTRI	CSPTRI	—	—	CHETRI	—
—	CHPTRI	—	SSYTRI_ROOK *	CSYTRI_ROOK *	—
SSPTRS	CSPTRS	—	—	CHETRI_ROOK *	—
—	CHPTRS	—	SSYTRI2^{+}	CSYTRI2^{+}	—
SSTEBZ	—	—	—	CHETRI2^{+}	—
SSTEDC	CSTEDC	O	SSYTRI2X^{+}	CSYTRI2X^{+}	—
SSTEGR	CSTEGR	—	—	CHETRI2X^{+}	—
SSTEIN	CSTEIN	—	SSYTRS	CSYTRS	O
SSTEQR	CSTEQR	—	—	CHETRS	—
SSTERF	—	—	SSYTRS_ROOK *	CSYTRS_ROOK *	—
SSTEV	—	—	—	CHETRS_ROOK *	—
SSTEVD	—	—	SSYTRS2^{+}	CSYTRS2^{+}	—
SSTEVR	—	—	—	CHETRS2^{+}	—
SSTEVX	—	—	STBCON	CTBCON	—
SSYCON	CSYCON	—	STBRFS	CTBRFS	—
—	CHECON	—	STBTRS	CTBTRS	—
SSYCON_ROOK *	CSYCON_ROOK *	—	SSTEMR	CSTEMR	—
—	CHECON_ROOK *	—	STFTRI	CTFTRI	—
SSYCONV^{+}	CSYCONV^{+}	—	STGEVC	CTGEVC	—
SSYEQUB	CSYEQUB	—	STGEXC	CTGEXC	—
SSYEV	CHEEV	—	STGSEN	CTGSEN	—
SSYEVD	CHEEVD	O	STGSJA	CTGSJA	—
SSYEVR	CHEEVR	—	STGSNA	CTGSNA	—
SSYEVX	CHEEVX	—	STGSYL	CTGSYL	—
SSYGST	CHEGST	—	STPCON	CTPCON	—
SSYGV	CHEGV	—	STPMQRT^{+}	CTPMQRT^{+}	—
SSYGVD	CHEGVD	O	STPQRT^{+}	CTPQRT^{+}	—
SSYGVX	CHEGVX	—	STPQRT2^{+}	CTPQRT2^{+}	—
SSYRFS	CSYRFS	O	STPRFS	CTPRFS	—
SSYRFSX	CSYRFSX	—	STPTRI	CTPTRI	—

Table A.9 Driver and Computational routines of LAPACK (continued)

Real routine	Complex routine	Parallel
STPTRS	CTPTRS	—
STRCON	CTRCON	—
STREVC	CTREVC	—
STREXC	CTREXC	—
STRRFS	CTRRFS	—
STRSEN	CTRSEN	—
STRSNA	CTRSNA	—
STRSYL	CTRSYL	—
STRTRI	CTRTRI	—
STRTRS	CTRTRS	—
STZRZF	CTZRZF	—

Table A.10 Auxiliary routines of LAPACK

Real routine	Complex routine
DLAT2S	—
—	CLACGV
—	CLACRM
—	CLACRT
—	CLAESY
—	CROT
—	CSPMV
—	CSPR
—	CSROT
—	CSYMV
—	CSYR
—	ICMAX1
ILACLC	—
ILACLR	—
ILADIAG	—
ILADLC	—
ILADLR	—
ILAENV	—
ILAPREC	—
ILASLC	—
ILASLR	—
ILATRANS	—
ILAUPLO	—
ILAZLC	—
ILAZLR	—
ILAVER	—
IPARMQ	—
LSAME	—
LSAMEN	—
—	SCSUM1
SGBTF2	CGBTF2
SGEBD2	CGEBD2
SGEHD2	CGEHD2
SGELQ2	CGELQ2

Table A.10 Auxiliary routines of LAPACK (continued)

Real routine	Complex routine
SGEQL2	CGEQL2
SGEQR2	CGEQR2
SGERQ2	CGERQ2
SGEQR2P	CGEQR2P
SGESC2	CGESC2
SGETC	CGETC2
SGETF2	CGETF2
SGSVJ0	—
SGSVJ1	—
SGTTS2	CGTTS2
SISNAN	—
SLABAD	—
SLABRD	CLABRD
SLACN2	CLACN2
SLACON	CLACON
SLACPY	CLACPY
SLADIV	CLADIV
SLAE2	—
SLAEBZ	—
SLAED0	CLAED0
SLAED1	—
SLAED2	—
SLAED3	—
SLAED4	—
SLAED5	—
SLAED6	—
SLAED7	CLAED7
SLAED8	CLAED8
SLAED9	—
SLAEDA	—
SLAEIN	CLAEIN
SLAEV2	CLAEV2
SLAEXC	—
SLAG2	—
SLAG2D	CLAG2Z
SLAGS2	—
SLAGTF	—
SLAGTM	CLAGTM
SLAGTS	—
SLAGV2	—
SLAHQR	CLAHQR
SLAHR2	CLAHR2

Table A.10 Auxiliary routines of LAPACK (continued)

Real routine	Complex routine
SLAHRD	CLAHRD
SLAIC1	CLAIC1
SLAISNAN	—
SLALN2	—
SLALS0	CLALS0
SLALSA	CLALSA
SLALSD	CLALSD
SLAMCH	—
SLAMRG	—
SLANEG	—
SLANGB	CLANGB
SLANGE	CLANGE
SLANGT	CLANGT
SLANHS	CLANHS
SLANSB	CLANSB
SLANSF	CLANHF
—	CLANHB
SLANSP	CLANSP
—	CLANHP
SLANST	CLANHT
SLANSY	CLANSY
—	CLANHE
SLANTB	CLANTB
SLANTP	CLANTP
SLANTR	CLANTR
SLANV2	—
SLAPLL	CLAPLL
SLAPMR^{+}	CLAPMR^{+}
SLAPMT	CLAPMT
SLAPY2	—
SLAPY3	—
SLAQGB	CLAQGB
SLAQGE	CLAQGE
SLAQP2	CLAQP2
SLAQPS	CLAQPS
SLAQR0	CLAQR0
SLAQR1	CLAQR1
SLAQR2	CLAQR2
SLAQR3	CLAQR3
SLAQR4	CLAQR4
SLAQR5	CLAQR5
SLAQSB	CLAQSB
SLAQSP	CLAQSP

Table A.10 Auxiliary routines of LAPACK (continued)

Real routine	Complex routine
SLAQSY	CLAQSY
SLAQTR	—
SLAR1V	CLAR1V
SLAR2V	CLAR2V
SLARF	CLARF
SLARFB	CLARFB
SLARFG	CLARFG
SLARFGP	CLARFGP
SLARFT	CLARFT
SLARFX	CLARFX
SLARGV	CLARGV
SLARNV	CLARNV
SLARRA	—
SLARRB	—
SLARRC	—
SLARRD	—
SLARRE	—
SLARRF	—
SLARRJ	—
SLARRK	—
SLARRR	—
SLARRV	CLARRV
SLARSCL2	CLARSCL2
SLASCL2	CLASCL2
SLARTG	CLARTG
SLARTGP^{+}	—
SLARTV	CLARTV
SLARUV	—
SLARZ	CLARZ
SLARZB	CLARZB
SLARZT	CLARZT
SLAS2	—
SLASCL	CLASCL
SLASD0	—
SLASD1	—
SLASD2	—
SLASD3	—
SLASD4	—
SLASD5	—
SLASD6	—
SLASD7	—
SLASD8	—
SLASD9	—
SLASDA	—
SLASDQ	—

Table A.10 Auxiliary routines of LAPACK (continued)

Real routine	Complex routine
SLASDT	—
SLASET	CLASET
SLASQ1	—
SLASQ2	—
SLASQ3	—
SLASQ4	—
SLASQ5	—
SLASQ6	—
SLASR	CLASR
SLASRT	—
SLASSQ	CLASSQ
SLASV2	—
SLASWP	CLASWP
SLASY2	—
SLASYF	CLASYF
—	CLAHEF
SLASYF_ROOK *	CLASYF_ROOK *
—	CLAHEF_ROOK *
SLATBS	CLATBS
SLATDF	CLATDF
SLATPS	CLATPS
SLATRD	CLATRD
SLATRS	CLATRS
SLATRZ	CLATRZ
SLAUU2	CLAUU2
SLAUUM	CLAUUM
SLAZQ3	—
SLAZQ4	—
SLA_GBAMV	CLA_GBAMV
SLA_GBRCOND	CLA_GBRCOND_C
—	CLA_GBRCOND_X
SLA_GBRFSX_EXTENDED	CLA_GBRFSX_EXTENDED
SLA_GBRPVGRW	CLA_GBRPVGRW
SLA_GEAMV	CLA_GEAMV
SLA_GERCOND	CLA_GERCOND_C
—	CLA_GERCOND_X
SLA_GERFSX_EXTENDED	CLA_GERFSX_EXTENDED
—	CLA_HEAMV
—	CLA_HERCOND_C
—	CLA_HERCOND_X
—	CLA_HERFSX_EXTENDED
—	CLA_HERPVGRW
SLA_LIN_BERR	CLA_LIN_BERR
SLA_PORCOND	CLA_PORCOND_C

Table A.10 Auxiliary routines of LAPACK (continued)

Real routine	Complex routine
—	CLA_PORCOND_X
SLA_PORFSX_EXTENDED	CLA_PORFSX_EXTENDED
SLA_PORPVGRW	CLA_PORPVGRW
SLA_RPVGRW	CLA_RPVGRW
SLA_SYAMV	CLA_SYAMV
SLA_SYRCOND	CLA_SYRCOND_C
—	CLA_SYRCOND_X
SLA_SYRFSX_EXTENDED	CLA_SYRFSX_EXTENDED
SLA_SYRPVGRW	CLA_SYRPVGRW
SLA_WWADDW	CLA_WWADDW
SORG2L	CUNG2L
SORG2R	CUNG2R
SORGL2	CUNGL2
SORGR2	CUNGR2
SORM2L	CUNM2L
SORM2R	CUNM2R
SORML2	CUNML2
SORMR2	CUNMR2
SORMR3	CUNMR3
SPBTF2	CPBTF2
SPOTF2	CPOTF2
SPTTS2	CPTTS2
SRSCL	CSRSCL
SSFRK	CHFRK
—	CHLA_TRANSTYPE
SSYGS2	CHEGS2
SSYSWAPR^{+}	CSYSWAPR^{+}
—	CHESWAPR^{+}
SSYTD2	CHETD2
SSYTF2	CSYTF2
—	CHETF2
STFSM	CTFSM
STFTTP	CTFTTP
STFTTR	CTFTTR
STGEX2	CTGEX2
STGSY2	CTGSY2
STPRFB^{+}	CTPRFB^{+}
STPTTF	CTPTTF
STPTTR	CTPTTR
STRTI2	CTRTI2
STRTTF	CTRTTF
STRTTP	CTRTTP
XERBLA	—
XERBLA_ARRAY	—

C inteface LAPACK routines are listed in Table A.11.

Table A.11 C interface of LAPACK routines

Real routine	Complex routine	Real routine	Complex routine
LAPACKE_ilaver *	—	LAPACKE_sgeqrf	LAPACKE_cgeqrf
LAPACKE_dsgesv	LAPACKE_zcgesv	LAPACKE_sgeqrfp	LAPACKE_cgeqrfp
LAPACKE_dsposv	LAPACKE_zcposv	LAPACKE_sgeqrt	LAPACKE_cgeqrt
LAPACKE_dtpqrt	LAPACKE_ctpqrt	LAPACKE_sgeqrt2	LAPACKE_cgeqrt2
LAPACKE_sbbcsd	LAPACKE_cbbcsd	LAPACKE_sgeqrt3	LAPACKE_cgeqrt3
LAPACKE_sbdsdc	—	LAPACKE_sgerfs	LAPACKE_cgerfs
LAPACKE_sbdsqr	LAPACKE_cbdsqr	LAPACKE_sgerfsx	LAPACKE_cgerfsx
LAPACKE_sdisna	—	LAPACKE_sgerqf	LAPACKE_cgerqf
LAPACKE_sgbbrd	LAPACKE_cgbbrd	LAPACKE_sgesdd	LAPACKE_cgesdd
LAPACKE_sgbcon	LAPACKE_cgbcon	LAPACKE_sgesv	LAPACKE_cgesv
LAPACKE_sgbequ	LAPACKE_cgbequ	LAPACKE_sgesvd	LAPACKE_cgesvd
LAPACKE_sgbequb	LAPACKE_cgbequb	LAPACKE_sgesvj	—
LAPACKE_sgbrfs	LAPACKE_cgbrfs	LAPACKE_sgesvx	LAPACKE_cgesvx
LAPACKE_sgbrfsx	LAPACKE_cgbrfsx	LAPACKE_sgesvxx	LAPACKE_cgesvxx
LAPACKE_sgbsv	LAPACKE_cgbsv	LAPACKE_sgetf2	LAPACKE_cgetf2
LAPACKE_sgbsvx	LAPACKE_cgbsvx	LAPACKE_sgetrf	LAPACKE_cgetrf
LAPACKE_sgbsvxx	LAPACKE_cgbsvxx	LAPACKE_sgetri	LAPACKE_cgetri
LAPACKE_sgbtrf	LAPACKE_cgbtrf	LAPACKE_sgetrs	LAPACKE_cgetrs
LAPACKE_sgbtrs	LAPACKE_cgbtrs	LAPACKE_sggbak	LAPACKE_cggbak
LAPACKE_sgebak	LAPACKE_cgebak	LAPACKE_sggbal	LAPACKE_cggbal
LAPACKE_sgebal	LAPACKE_cgebal	LAPACKE_sgges	LAPACKE_cgges
LAPACKE_sgebrd	LAPACKE_cgebrd	LAPACKE_sggesx	LAPACKE_cggesx
LAPACKE_sgecon	LAPACKE_cgecon	LAPACKE_sggev	LAPACKE_cggev
LAPACKE_sgeequ	LAPACKE_cgeequ	LAPACKE_sggevx	LAPACKE_cggevx
LAPACKE_sgeequb	LAPACKE_cgeequb	LAPACKE_sggglm	LAPACKE_cggglm
LAPACKE_sgees	LAPACKE_cgees	LAPACKE_sgghrd	LAPACKE_cgghrd
LAPACKE_sgeesx	LAPACKE_cgeesx	LAPACKE_sgglse	LAPACKE_cgglse
LAPACKE_sgeev	LAPACKE_cgeev	LAPACKE_sggqrf	LAPACKE_cggqrf
LAPACKE_sgeevx	LAPACKE_cgeevx	LAPACKE_sggrqf	LAPACKE_cggrqf
LAPACKE_sgehrd	LAPACKE_cgehrd	LAPACKE_sggsvd	LAPACKE_cggsvd
LAPACKE_sgejsv	—	LAPACKE_sggsvp	LAPACKE_cggsvp
LAPACKE_sgelq2	LAPACKE_cgelq2	LAPACKE_sgtcon	LAPACKE_cgtcon
LAPACKE_sgelqf	LAPACKE_cgelqf	LAPACKE_sgtrfs	LAPACKE_cgtrfs
LAPACKE_sgels	LAPACKE_cgels	LAPACKE_sgtsv	LAPACKE_cgtsv
LAPACKE_sgelsd	LAPACKE_cgelsd	LAPACKE_sgtsvx	LAPACKE_cgtsvx
LAPACKE_sgelss	LAPACKE_cgelss	LAPACKE_sgttrf	LAPACKE_cgttrf
LAPACKE_sgelsy	LAPACKE_cgelsy	LAPACKE_sgttrs	LAPACKE_cgttrs
LAPACKE_sgemqrt	LAPACKE_cgemqrt	—	LAPACKE_chfrk
LAPACKE_sgeqlf	LAPACKE_cgeqlf	LAPACKE_shgeqz	LAPACKE_chgeqz
LAPACKE_sgeqp3	LAPACKE_cgeqp3	LAPACKE_shsein	—
LAPACKE_sgeqpf	LAPACKE_cgeqpf	LAPACKE_shseqr	LAPACKE_chseqr
LAPACKE_sgeqr2	LAPACKE_cgeqr2	—	LAPACKE_chsein

Table A.11 C interface of LAPACK routines (continued)

Real routine	Complex routine
—	LAPACKE_clacgv
LAPACKE_slacpy	LAPACKE_clacpy
LAPACKE_slag2d	LAPACKE_clag2z
LAPACKE_slamch	—
LAPACKE_slange	LAPACKE_clange
LAPACKE_slansy	LAPACKE_clansy
—	LAPACKE_clanhe
LAPACKE_slantr	LAPACKE_clantr
LAPACKE_slapmr	LAPACKE_clapmr
LAPACKE_slapy2	—
LAPACKE_slapy3	—
LAPACKE_slarfb	LAPACKE_clarfb
LAPACKE_slarfg	LAPACKE_clarfg
LAPACKE_slarft	LAPACKE_clarft
LAPACKE_slarfx	LAPACKE_clarfx
LAPACKE_slarnv	LAPACKE_clarnv
LAPACKE_slartgp	—
LAPACKE_slartgs	—
LAPACKE_slaset	LAPACKE_claset
LAPACKE_slasrt	—
LAPACKE_slaswp	LAPACKE_claswp
LAPACKE_slauum	LAPACKE_clauum
LAPACKE_sopgtr	—
LAPACKE_sopmtr	—
LAPACKE_sorbdb	LAPACKE_cunbdb
LAPACKE_sorcsd	LAPACKE_cuncsd
LAPACKE_sorgbr	LAPACKE_cungbr
LAPACKE_sorghr	LAPACKE_cunghr
LAPACKE_sorglq	LAPACKE_cunglq
LAPACKE_sorgql	LAPACKE_cungql
LAPACKE_sorgqr	LAPACKE_cungqr
LAPACKE_sorgrq	LAPACKE_cungrq
LAPACKE_sorgtr	LAPACKE_cungtr
LAPACKE_sormbr	LAPACKE_cunmbr
LAPACKE_sormhr	LAPACKE_cunmhr
LAPACKE_sormlq	LAPACKE_cunmlq
LAPACKE_sormql	LAPACKE_cunmql
LAPACKE_sormqr	LAPACKE_cunmqr
LAPACKE_sormrq	LAPACKE_cunmrq
LAPACKE_sormrz	LAPACKE_cunmrz
LAPACKE_sormtr	LAPACKE_cunmtr
LAPACKE_spbcon	LAPACKE_cpbcon
LAPACKE_spbequ	LAPACKE_cpbequ
LAPACKE_spbrfs	LAPACKE_cpbrfs
LAPACKE_spbstf	LAPACKE_cpbstf
LAPACKE_spbsv	LAPACKE_cpbsv
LAPACKE_spbsvx	LAPACKE_cpbsvx
LAPACKE_spbtrf	LAPACKE_cpbtrf
LAPACKE_spbtrs	LAPACKE_cpbtrs
LAPACKE_spftrf	LAPACKE_cpftrf
LAPACKE_spftri	LAPACKE_cpftri
LAPACKE_spftrs	LAPACKE_cpftrs
LAPACKE_spocon	LAPACKE_cpocon
LAPACKE_spoequ	LAPACKE_cpoequ
LAPACKE_spoequb	LAPACKE_cpoequb
LAPACKE_sporfs	LAPACKE_cporfs
LAPACKE_sporfsx	LAPACKE_cporfsx
LAPACKE_sposv	LAPACKE_cposv
LAPACKE_sposvx	LAPACKE_cposvx
LAPACKE_sposvxx	LAPACKE_cposvxx
LAPACKE_spotrf	LAPACKE_cpotrf
LAPACKE_spotri	LAPACKE_cpotri
LAPACKE_spotrs	LAPACKE_cpotrs
LAPACKE_sppcon	LAPACKE_cppcon
LAPACKE_sppequ	LAPACKE_cppequ
LAPACKE_spprfs	LAPACKE_cpprfs
LAPACKE_sppsv	LAPACKE_cppsv
LAPACKE_sppsvx	LAPACKE_cppsvx
LAPACKE_spptrf	LAPACKE_cpptrf
LAPACKE_spptri	LAPACKE_cpptri
LAPACKE_spptrs	LAPACKE_cpptrs
LAPACKE_spstrf	LAPACKE_cpstrf
LAPACKE_sptcon	LAPACKE_cptcon
LAPACKE_spteqr	LAPACKE_cpteqr
LAPACKE_sptrfs	LAPACKE_cptrfs
LAPACKE_sptsv	LAPACKE_cptsv
LAPACKE_sptsvx	LAPACKE_cptsvx
LAPACKE_spttrf	LAPACKE_cpttrf
LAPACKE_spttrs	LAPACKE_cpttrs
LAPACKE_ssbev	LAPACKE_chbev
LAPACKE_ssbevd	LAPACKE_chbevd
LAPACKE_ssbevx	LAPACKE_chbevx
LAPACKE_ssbgst	LAPACKE_chbgst
LAPACKE_ssbgv	LAPACKE_chbgv
LAPACKE_ssbgvd	LAPACKE_chbgvd
LAPACKE_ssbgvx	LAPACKE_chbgvx
LAPACKE_ssbtrd	LAPACKE_chbtrd
LAPACKE_ssfrk	—
LAPACKE_sspcon	LAPACKE_cspcon
—	LAPACKE_chpcon

Table A.11 C interface of LAPACK routines (continued)

Real routine	Complex routine
LAPACKE_sspev	LAPACKE_chpev
LAPACKE_sspevd	LAPACKE_chpevd
LAPACKE_sspevx	LAPACKE_chpevx
LAPACKE_sspgst	LAPACKE_chpgst
LAPACKE_sspgv	LAPACKE_chpgv
LAPACKE_sspgvd	LAPACKE_chpgvd
LAPACKE_sspgvx	LAPACKE_chpgvx
LAPACKE_ssprfs	LAPACKE_csprfs
—	LAPACKE_chprfs
LAPACKE_sspsv	LAPACKE_cspsv
—	LAPACKE_chpsv
LAPACKE_sspsvx	LAPACKE_cspsvx
—	LAPACKE_chpsvx
LAPACKE_ssptrd	LAPACKE_chptrd
LAPACKE_ssptrf	LAPACKE_csptrf
—	LAPACKE_chptrf
LAPACKE_ssptri	LAPACKE_csptri
—	LAPACKE_chptri
LAPACKE_ssptrs	LAPACKE_csptrs
—	LAPACKE_chptrs
LAPACKE_sstebz	—
LAPACKE_sstedc	LAPACKE_cstedc
LAPACKE_sstegr	LAPACKE_cstegr
LAPACKE_sstein	LAPACKE_cstein
LAPACKE_sstemr	LAPACKE_cstemr
LAPACKE_ssteqr	LAPACKE_csteqr
LAPACKE_ssterf	—
LAPACKE_sstev	—
LAPACKE_sstevd	—
LAPACKE_sstevr	—
LAPACKE_sstevx	—
LAPACKE_ssycon	LAPACKE_csycon
—	LAPACKE_checon
LAPACKE_ssyconv	LAPACKE_csyconv
LAPACKE_ssyequb	LAPACKE_csyequb
—	LAPACKE_cheequb
LAPACKE_ssyev	LAPACKE_cheev
LAPACKE_ssyevd	LAPACKE_cheevd
LAPACKE_ssyevr	LAPACKE_cheevr
LAPACKE_ssyevx	LAPACKE_cheevx
LAPACKE_ssygst	LAPACKE_chegst
LAPACKE_ssygv	LAPACKE_chegv
LAPACKE_ssygvd	LAPACKE_chegvd
LAPACKE_ssygvx	LAPACKE_chegvx

Real routine	Complex routine
—	LAPACKE_csyr
LAPACKE_ssyrfs	LAPACKE_csyrfs
—	LAPACKE_cherfs
LAPACKE_ssyrfsx	LAPACKE_csyrfsx
—	LAPACKE_cherfsx
LAPACKE_ssysv	LAPACKE_csysv
—	LAPACKE_chesv
LAPACKE_ssysv_rook *	LAPACKE_csysv_rook *
LAPACKE_ssysvx	LAPACKE_csysvx
—	LAPACKE_chesvx
LAPACKE_ssysvxx	LAPACKE_csysvxx
—	LAPACKE_chesvxx
LAPACKE_ssyswapr	LAPACKE_csyswapr
—	LAPACKE_cheswapr
LAPACKE_ssytrd	LAPACKE_chetrd
LAPACKE_ssytrf	LAPACKE_csytrf
—	LAPACKE_chetrf
LAPACKE_ssytri	LAPACKE_csytri
—	LAPACKE_chetri
LAPACKE_ssytri2	LAPACKE_csytri2
—	LAPACKE_chetri2
LAPACKE_ssytri2x	LAPACKE_csytri2x
—	LAPACKE_chetri2x
LAPACKE_ssytrs	LAPACKE_csytrs
—	LAPACKE_chetrs
LAPACKE_ssytrs2	LAPACKE_csytrs2
—	LAPACKE_chetrs2
LAPACKE_stbcon	LAPACKE_ctbcon
LAPACKE_stbrfs	LAPACKE_ctbrfs
LAPACKE_stbtrs	LAPACKE_ctbtrs
LAPACKE_stfsm	LAPACKE_ctfsm
LAPACKE_stftri	LAPACKE_ctftri
LAPACKE_stfttp	LAPACKE_ctfttp
LAPACKE_stfttr	LAPACKE_ctfttr
LAPACKE_stgevc	LAPACKE_ctgevc
LAPACKE_stgexc	LAPACKE_ctgexc
LAPACKE_stgsen	LAPACKE_ctgsen
LAPACKE_stgsja	LAPACKE_ctgsja
LAPACKE_stgsna	LAPACKE_ctgsna
LAPACKE_stgsyl	LAPACKE_ctgsyl
LAPACKE_stpcon	LAPACKE_ctpcon
LAPACKE_stpmqrt	LAPACKE_ctpmqrt
LAPACKE_stpqrt2	LAPACKE_ctpqrt2
LAPACKE_stprfb	LAPACKE_ctprfb

Table A.11 C interface of LAPACK routines (continued)

Real routine	Complex routine
LAPACKE_stprfs	LAPACKE_ctprfs
LAPACKE_stptri	LAPACKE_ctptri
LAPACKE_stptrs	LAPACKE_ctptrs
LAPACKE_stpttf	LAPACKE_ctpttf
LAPACKE_stpttr	LAPACKE_ctpttr
LAPACKE_strcon	LAPACKE_ctrcon
LAPACKE_strevc	LAPACKE_ctrevc
LAPACKE_strexc	LAPACKE_ctrexc
LAPACKE_strrfs	LAPACKE_ctrrfs
LAPACKE_strsen	LAPACKE_ctrsen

Real routine	Complex routine
LAPACKE_strsna	LAPACKE_ctrsna
LAPACKE_strsyl	LAPACKE_ctrsyl
LAPACKE_strtri	LAPACKE_ctrtri
LAPACKE_strtrs	LAPACKE_ctrtrs
LAPACKE_strttf	LAPACKE_ctrttf
LAPACKE_strttp	LAPACKE_ctrttp
LAPACKE_stzrzf	LAPACKE_ctzrzf
—	LAPACKE_cupgtr
—	LAPACKE_cupmtr

PLASMA routines are listed in Table A.12 and Table A.13. They are same routine name in C and Fortran. The symbol * shows routines supported in C interface only.

Table A.12 PLASMA Simple interface routines

Real routine	Complex routine
PLASMA_sgebrd	PLASMA_cgebrd
PLASMA_sgecfi	PLASMA_cgecfi
PLASMA_sgecfi_Async	PLASMA_cgecfi_Async
PLASMA_sgecon	PLASMA_cgecon
PLASMA_sgelqf	PLASMA_cgelqf
PLASMA_sgelqs	PLASMA_cgelqs
PLASMA_sgels	PLASMA_cgels
PLASMA_sgemm	PLASMA_cgemm
PLASMA_sgeqp3	PLASMA_cgeqp3
PLASMA_sgeqrf	PLASMA_cgeqrf
PLASMA_sgeqrs	PLASMA_cgeqrs
PLASMA_sgesdd	PLASMA_cgesdd
PLASMA_sgesv	PLASMA_cgesv
PLASMA_sgesv_incpiv	PLASMA_cgesv_incpiv
PLASMA_sgesvd	PLASMA_cgesvd
PLASMA_sgetmi	PLASMA_cgetmi
PLASMA_sgetmi_Async	PLASMA_cgetmi_Async
PLASMA_sgetrf	PLASMA_cgetrf
PLASMA_sgetrf_incpiv	PLASMA_cgetrf_incpiv
PLASMA_sgetrf_nopiv	PLASMA_cgetrf_nopiv
PLASMA_sgetrf_tntpiv	PLASMA_cgetrf_tntpiv
PLASMA_sgetri	PLASMA_cgetri
PLASMA_sgetrs	PLASMA_cgetrs

Table A.12 PLASMA Simple interface routines (continued)

Real routine	Complex routine
PLASMA_sgetrs_incpiv	PLASMA_cgetrs_incpiv
PLASMA_slacpy	PLASMA_clacpy
PLASMA_slange	PLASMA_clange
PLASMA_slansy	PLASMA_clansy
—	PLASMA_clanhe
PLASMA_slantr	PLASMA_clantr
PLASMA_slaset	PLASMA_claset
PLASMA_slaswp	PLASMA_claswp
PLASMA_slaswpc	PLASMA_claswpc
PLASMA_slauum	PLASMA_clauum
PLASMA_sorglq	PLASMA_cunglq
PLASMA_sorgqr	PLASMA_cungqr
PLASMA_sormlq	PLASMA_cunmlq
PLASMA_sormqr	PLASMA_cunmqr
PLASMA_splgsy	PLASMA_cplgsy
—	PLASMA_cplghe
PLASMA_splrnt	PLASMA_cplrnt
PLASMA_spltmg	PLASMA_cpltmg
PLASMA_spocon	PLASMA_cpocon
PLASMA_sposv	PLASMA_cposv
PLASMA_spotrf	PLASMA_cpotrf
PLASMA_spotri	PLASMA_cpotri
PLASMA_spotrs	PLASMA_cpotrs
PLASMA_ssyev	PLASMA_cheev
PLASMA_ssyevd	PLASMA_cheevd
PLASMA_ssyevr	PLASMA_cheevr
PLASMA_ssygst	PLASMA_chegst
PLASMA_ssygv	PLASMA_chegv
PLASMA_ssygvd	PLASMA_chegvd
PLASMA_ssymm	PLASMA_csymm
—	PLASMA_chemm
PLASMA_ssyr2k	PLASMA_csyr2k
—	PLASMA_cher2k
PLASMA_ssyrk	PLASMA_csyrk
—	PLASMA_cherk
PLASMA_ssytrd	PLASMA_chetrd
PLASMA_strmm	PLASMA_ctrmm
PLASMA_strsm	PLASMA_ctrsm
PLASMA_strsmpl	PLASMA_ctrsmpl
PLASMA_strsmrv	PLASMA_ctrsmrv
PLASMA_strtri	PLASMA_ctrtri
PLASMA_dsgesv	PLASMA_zcgesv
PLASMA_dsposv	PLASMA_zcposv
PLASMA_dsungesv	PLASMA_zcungesv

Table A.13 PLASMA Main Routines

Real routine	Complex routine
PLASMA_Alloc_Workspace_sgebrd	PLASMA_Alloc_Workspace_cgebrd
PLASMA_Alloc_Workspace_sgeev	PLASMA_Alloc_Workspace_cgeev
PLASMA_Alloc_Workspace_sgehrd	PLASMA_Alloc_Workspace_cgehrd
PLASMA_Alloc_Workspace_sgels	PLASMA_Alloc_Workspace_cgels
PLASMA_Alloc_Workspace_sgeqrf	PLASMA_Alloc_Workspace_cgeqrf
PLASMA_Alloc_Workspace_sgelqf	PLASMA_Alloc_Workspace_cgelqf
PLASMA_Alloc_Workspace_sgesdd	PLASMA_Alloc_Workspace_cgesdd
PLASMA_Alloc_Workspace_sgesv_incpiv	PLASMA_Alloc_Workspace_cgesv_incpiv
PLASMA_Alloc_Workspace_sgesvd	PLASMA_Alloc_Workspace_cgesvd
PLASMA_Alloc_Workspace_sgetrf_incpiv	PLASMA_Alloc_Workspace_cgetrf_incpiv
PLASMA_Alloc_Workspace_ssyev	PLASMA_Alloc_Workspace_cheev
PLASMA_Alloc_Workspace_ssyevd	PLASMA_Alloc_Workspace_cheevd
PLASMA_Alloc_Workspace_ssyevr	PLASMA_Alloc_Workspace_cheevr
PLASMA_Alloc_Workspace_ssygv	PLASMA_Alloc_Workspace_chegv
PLASMA_Alloc_Workspace_ssygvd	PLASMA_Alloc_Workspace_chegvd
PLASMA_Alloc_Workspace_ssytrd	PLASMA_Alloc_Workspace_chetrd
PLASMA_Dealloc_Handle	—
PLASMA_Desc_Create	—
PLASMA_Desc_Destroy	—
PLASMA_Disable	—
PLASMA_Enable	—
PLASMA_Finalize	—
PLASMA_Get	—
PLASMA_Init	—
PLASMA_Init_Affinity *	—
PLASMA_Sequence_Create *	—
PLASMA_Sequence_Destroy *	—
PLASMA_Sequence_Flush *	—
PLASMA_Sequence_Wait *	—
PLASMA_Set	—
PLASMA_Version	—

Old routines in Table A.14 are replaced with new routines from LAPACK version 3.0 and from LAPACK version 3.2.2. The old routines are still included in the library but the user advised to upgrade to the new routines.

Table A.14 Old routines included in the library

Old routines	New routines
#GEGS	#GGES
#GEGV	#GGEV
#GELSX	#GELSY
#GEQPF	#GEQP3

Old routines	New routines
#TZRQF	#TZRZF
#LATZM	#ORMRZ, #UNMRZ
#LAZQ3 *	#LASQ3 *
#LAZQ4 *	#LASQ4 *

The interfaces of the routines in Table A.15 and Table A.16 are changed from LAPACK version 3.1.1 and from LAPACK version 3.2.2.

User program, which calls these routines, have to be changed for new interface.

Table A.15 The interface change for this LAPACK 3.1.1

Real routine	Complex routine
SLAR1V	—
SLARRB	—
SLARRE	—

Real routine	Complex routine
SLARRF	—
SLARRV	—

Table A.16 The interface change for this LAPACK 3.2

Real routine	Complex routine
—	ZCGESV
SLASQ3	—
SLASQ4	—

A.3 ScaLAPACK

A.3.1 ScaLAPACK

The routines that are supplied with ScaLAPACK are listed in Table A.17 and Table A.18. Table A.17 lists driver routines and computational routines. Table A.18 lists auxiliary routines.

The routine names in the following tables are the names of real and complex routines. The character indicating the supported precision is the second character of the routine name. For a double precision routine, replace the “S” of the real routine with “D”. For a double complex routine, replace the “C” of the complex routine with “Z”.

Auxiliary routine PXERBLA is unique and do not depend on data type.

The symbol + shows new routines in version 1.8 of ScaLAPACK on Netlib.

The symbol * shows new routines in version 2.0 of ScaLAPACK on Netlib.

Table A.17 Driver and Computational routines of ScaLAPACK

Real routine	Complex routine	Real routine	Complex routine
BSTREXC *	—	PSHSEQR *	—
PSDBSV	PCDBSV	PSORGLQ	PCUNGLQ
PSDBTRF	PCDBTRF	PSORGQL	PCUNGQL
PSDBTRS	PCDBTRS	PSORGQR	PCUNGQR
PSDTSV	PCDTSV	PSORGRQ	PCUNGRQ
PSDTTRF	PCDTTRF	PSORMBR	PCUNMBR
PSDTTRS	PCDTTRS	PSORMHR	PCUNMHR
PSGBSV	PCGBSV	PSORMLQ	PCUNMLQ
PSGBTRF	PCGBTRF	PSORMQL	PCUNMQL
PSGBTRS	PCGBTRS	PSORMQR	PCUNMQR
PSGEBAL *	—	PSORMRQ	PCUNMRQ
PSGEBRD	PCGEBRD	PSORMRZ	PCUNMRZ
PSGECON	PCGECON	PSORMTR	PCUNMTR
PSGEEQU	PCGEEQU	PSPBSV	PCPBSV
PSGEHRD	PCGEHRD	PSPBTRF	PCPBTRF
PSGELQF	PCGELQF	PSPBTRS	PCPBTRS
PSGELS	PCGELS	PSPOCON	PCPOCON
PSGEQLF	PCGEQLF	PSPOEQU	PCPOEQU
PSGEQPF	PCGEQPF	PSPORFS	PCPORFS
PSGEQRF	PCGEQRF	PSPOSV	PCPOSV
PSGERFS	PCGERFS	PSPOSVX	PCPOSVX
PSGERQF	PCGERQF	PSPOTRF	PCPOTRF
PSGESV	PCGESV	PSPOTRI	PCPOTRI
PSGESVD	PCGESVD+	PSPOTRS	PCPOTRS
PSGESVX	PCGESVX	PSPTSV	PCPTSV
PSGETRF	PCGETRF	PSPTTRF	PCPTTRF
PSGETRI	PCGETRI	PSPTTRS	PCPTTRS
PSGETRS	PCGETRS	PSSTEBZ	—
PSGGQRF	PCGGQRF	PSSTEDC	—
PSGGRQF	PCGGRQF	PSSTEIN	PCSTEIN

Table A.17 Driver and Computational routines of ScaLAPACK (continued)

Real routine	Complex routine
PSSYEV	—
PSSYEVD	PCHEEVD
PSSYEVR *	PCHEEVR *
PSSYEVX	PCHEEVX
PSSYGST	PCHEGST
PSSYGVX	PCHEGVX
PSSYNTRD	PCHENTRD
PSSYNGST	PCHENGST
PSSYTRD	PCHETRD

Real routine	Complex routine
PSSYTTRD	PCHETTRD
PSTRCON	PCTRCON
—	PCTREVC
PSTRORD *	—
PSTRRFS	PCTRRFS
PSTRSEN *	—
PSTRTRI	PCTRTRI
PSTRTRS	PCTRTRS
PSTZRZF	PCTZRZF

Table A.18 Auxiliary routines of ScaLAPACK

Real routine	Complex routine
BSLAAPP *	—
BSLAEXC *	—
—	PCLACGV
—	PCMAX1
—	PSCSUM1
PILAENVX *	—
PILAVER *	—
PIPARMQ *	—
PMPCOL *	—
PMPIM2 *	—
PSDBTRSV	PCDBTRSV
PSDTTRSV	PCDTTRSV
PSGEBD2	PCGEBD2
PSGEHD2	PCGEHD2
PSGELQ2	PCGELQ2
PSGEQL2	PCGEQL2
PSGEQR2	PCGEQR2
PSGERQ2	PCGERQ2
PSGETF2	PCGETF2
PSLABAD	—
PSLABRD	PCLABRD
PSLACHKIEEE	—
PSLACON	PCLACON
PSLACONSB	PCLACONSB
PSLACP2	PCLACP2
PSLACP3	PCLACP3
PSLACPY	PCLACPY
PSLAED0	—
PSLAED1	—
PSLAED2	—
PSLAED3	—
PSLAEDZ	—

Real routine	Complex routine
PSLAEVSWP	PCLAEVSWP
PSLAHQR	PCLAHQR
PSLAHRD	PCLAHRD
PSLAIECT (*1)	—
PSLAMCH	—
PSLAMR1D	PCLAMR1D
PSLAMVE *	—
PSLANGE	PCLANGE
PSLANHS	PCLANHS
PSLANSY	PCLANSY
—	PCLANHE
PSLANTR	PCLANTR
PSLAPIV	PCLAPIV
PSLAQGE	PCLAQGE
PSLAQR0 *	—
PSLAQR1 *	—
PSLAQR2 *	—
PSLAQR3 *	—
PSLAQR4 *	—
PSLAQR5 *	—
PSLAQSY	PCLAQSY
PSLARED1D	—
PSLARED2D	—
PSLARF	PCLARF
PSLARFB	PCLARFB
—	PCLARFC
PSLARFG	PCLARFG
PSLARFT	PCLARFT
PSLARZ	PCLARZ
PSLARZB	PCLARZB
—	PCLARZC
PSLARZT	PCLARZT

Table A.18 Auxiliary routines of ScaLAPACK (continued)

Real routine	Complex routine
PSLASCL	PCLASCL
PSLASET	PCLASET
PSLASMSUB	PCLASMSUB
PSLASNBT	—
PSLASSQ	PCLASSQ
PSLASRT	—
PSLASWP	PCLASWP
PSLATRA	PCLATRA
PSLATRD	PCLATRD
PSLATRS	PCLATRS
PSLATRZ	PCLATRZ
—	PCLATTRS
PSLAUU2	PCLAUU2
PSLAUUM	PCLAUUM
PSLAWIL	PCLAWIL
PSORG2L	PCUNG2L
PSORG2R	PCUNG2R
PSORGL2	PCUNGL2
PSORGR2	PCUNGR2
PSORM2L	PCUNM2L
PSORM2R	PCUNM2R
PSORML2	PCUNML2
PSORMR2	PCUNMR2
PSPBTRSV	PCPBTRSV
PSPTTRSV	PCPTTRSV
PSPOTF2	PCPOTF2
PSROT *	—
PSRSCL	PCSRSCL(*2)

Real routine	Complex routine
PSSYGS2	PCHEGS2
PSSYTD2	PCHETD2
PSTRTI2	PCTRTI2
PXERBLA	—
SDBTF2	CDBTF2
SDBTRF	CDBTRF
SDTTRF	CDTTRF
SDTTRSV	CDTTRSV
SLAMOV *	CLAMOV *
SLAMSH	—
SLAQR6 *	—
SLAR1VA *	—
SLAREF	—
SLARRB2 *	—
SLARRD2 *	—
SLARRE2 *	—
SLARRE2A *	—
SLARRF2 *	—
SLARRV2 *	—
SLASORTE	—
SLASRT2	—
SPTTRSV	CPTTRSV
SSTEGR2 *	—
SSTEGR2A *	—
SSTEGR2B *	—
SSTEIN2	—
SSTEQR2	—

(*1) The double precision version of PSLAIECT is named PDLAIECTB
(*2) The double complex precision version of PCSRSCL is named PZDRSCL.

A.3.2 PBLAS

The PBLAS routines supplied with this software are listed in Table A.19 to Table A.21.

The mark # means that it takes either of:

S : REAL
D : DOUBLE PRECISION
C : COMPLEX
Z : COMPLEX*16

A combination of precisions means to use more than one precision. For example, “SC” of PSCNRM2 is a function returning real and complex entries.

Table A.19 Level 1 PBLAS routines

Routine name	Supported precision
P#SWAP	S, D, C, Z
P#SCAL	S, D, C, Z, CS, ZD
P#COPY	S, D, C, Z
P#AXPY	S, D, C, Z
P#DOT	S, D
P#DOTU	C, Z
P#DOTC	C, Z
P#NRM2	S, D, SC, DZ
P#ASUM	S, D, SC, DZ
P#AMAX	S, D, C, Z

Table A.20 Level 2 PBLAS routines

Routine name	Supported precision
P#GEMV	S, D, C, Z
P#HEMV	C, Z
P#SYMV	S, D
P#TRMV	S, D, C, Z
P#TRSV	S, D, C, Z
P#GER	S, D
P#GERU	C, Z
P#GERC	C, Z
P#HER	C, Z
P#HER2	C, Z
P#SYR	S, D
P#SYR2	S, D

Table A.21 Level 3 PBLAS routines

Routine name	Supported precision
P#GEMM	S, D, C, Z
P#SYMM	S, D, C, Z
P#HEMM	C, Z
P#SYRK	S, D, C, Z
P#HERK	C, Z
P#SYR2K	S, D, C, Z
P#HER2K	C, Z
P#TRAN	S, D
P#TRANU	C, Z
P#TRANC	C, Z
P#TRMM	S, D, C, Z
P#TRSM	S, D, C, Z

A.3.3 BLACS

The BLACS routines supplied with this software are listed in Table A.22

The mark # means that it takes either of:

S : REAL
D : DOUBLE PRECISION
C : COMPLEX
Z : COMPLEX*16
I : INTEGER

Table A.22 BLACS routines

Type	Routine name	Supported precision
Initialization	BLACS_PINFO BLACS_SETUP BLACS_GET BLACS_SET BLACS_GRIDINIT BLACS_GRIDMAP	—
Destruction	BLACS_FREEBUFF BLACS_GRIDEXIT BLACS_ABORT BLACS_EXIT	—
Sending	#GESD2D #GEBS2D #TRSD2D #TRBS2D	S, D, C, Z, I
Receiving	#GERV2D #GEBR2D #TRRV2D #TRBR2D	S, D, C, Z, I
Combine Operations	#GAMX2D #GAMN2D #GSUM2D	S, D, C, Z, I
Informational and Miscellaneous	BLACS_GRIDINFO BLACS_PNUM BLACS_PCOORD BLACS_BARRIER DCPUTIME00 DWALLTIME00 KSENDID KRECVID KBSID KBRID blacs2sys_handle sys2blacs_handle free_blacs_system_handle	—

A.3.4 TOOLS Routines

TOOLS and matrix redistribution routines are listed in Table A.23.

The mark # means that it takes either of:

S : REAL
D : DOUBLE PRECISION
C : COMPLEX
Z : COMPLEX*16
I : INTEGER

Table A.23 Tools routines

Routine name	Supported precision	Description
SL_INIT DESCINIT	—	TOOLS routine
P#GEMR2D P#TRMR2D	S, D, C, Z, I	Matrix Redistribution
P#LAREAD	S, D, C, Z	Read a matrix from a file and distribute it to the process grid
P#LAWRITE	S, D, C, Z	Write distribute matrix to a file

A.4 BLAS-like extensions

There are extended functions in addition to standard BLAS routines in this product.

A.4.1 Half precision BLAS routines

This product supports half precision routines for some BLAS routines. The parameter sequences are the same as double or single precision BLAS routines. Supported routines are listed in Table A.24.

Table A.24 Half precision BLAS routines

	Fortran BLAS routine name	CBLAS routine name	Function
Level 1	FJBLAS_SWAP_R16	fjcblas_swap_r16	$x \Leftrightarrow y$
	FJBLAS_SCAL_R16	fjcblas_scal_r16	$x \leftarrow ax$
	FJBLAS_COPY_R16	fjcblas_copy_r16	$y \leftarrow x$
	FJBLAS_AXPY_R16	fjcblas_axpy_r16	$y \leftarrow ax+y$
	FJBLAS_DOT_R16	fjcblas_dot_r16	dot$\leftarrow x^{\mathrm{T}}y$
	FJBLAS_ASUM_R16	fjcblas_asum_r16	asum$\leftarrow \|\|\mathrm{re}(x)\|\|_1+\|\|\mathrm{im}(x)\|\|_1$
	FJBLAS_AMAX_I32_R16	fjcblas_amax_i32_r16	amax$\leftarrow$argmax x_k
Level 2	FJBLAS_GEMV_R16	fjcblas_gemv_r16	$y \leftarrow aAx+by$
	FJBLAS_GER_R16	fjcblas_ger_r16	$A \leftarrow axy^{\mathrm{T}}+A$
Level 3	FJBLAS_GEMM_R16	fjcblas_gemm_r16	$C \leftarrow aAB+bC$

A.4.2 Sequential BLAS routines in the thread-parallel BLAS library

The sequential BLAS routines can be called in a user program which uses thread parallel BLAS routines. For example, a program, which calls thread parallel BLAS routines and link with thread parallel version of BLAS library, can calls sequential version of BLAS routines in parallel regions.

The dedicated BLAS routines for sequential use are provided corresponding to not parallelized routines as well as parallelized routines.

Their routine names are not the same as general BLAS routines to be able to call both of the routines in one program. The parameter sequences are the same as general BLAS routines.

The sequential routine names of Fortran interface BLAS add “L_” after the first letter of Netlib BLAS routine names.

Example 1:

```
CALL DGEMM(TRANSA, TRANSB, M, N, K, ...)
CALL DL_GEMM(TRANSA, TRANSB, M, N, K, ...)   ! sequential routine
```

Example 2:

```
I=IDAMAX(N, DX, INCX)
I=IL_DAMAX(N, DX, INCX)   ! sequential routine
```

The sequential routine names of Fortran interface half precision BLAS have the prefix “SS_” to half precision BLAS routine names.

Example 3:

```
CALL FJBLAS_GEMM_R16(TRANSA, TRANSB, M, N, K, …)
CALL SS_FJBLAS_GEMM_R16(TRANSA, TRANSB, M, N, K, …)   ! sequential routine
```

The sequential routine names for CBLAS has the prefix “#l_” to Netlib CBLAS routine names. # is 7th character of CBLAS routine name.

Example 4:

```
cblas_dgemm(layout, transa, transb, … )
dl_cblas_dgemm(layout, transa, transb, …)    // sequential routine
```

Example 5:

```
cblas_idamax(n, x, incx)
il_cblas_idamax(n, x, incx)    // sequential routine
```

The sequential routine names of half precision CBLAS and matrix copy/transpose routines have the prefix “ss_” to original routine names.

Example 6:

```
fjcblas_gemm_r16(layout, transa, transb, …)
ss_fjcblas_gemm_r16(layout, transa, transb, …)    // sequential routine
```

The sequential BLAS routines supplied with this software are listed in Table A.25 to Table A.29.

Table A.25 Level 1 sequential BLAS routines

Fortran BLAS sequential routine name	CBLAS sequential routine name	Supported precision (small letter for CBLAS)
#L_ROTG	#l_cblas_#rotg	S, D
#L_ROTMG	#l_cblas_#rotmg	S, D
#L_ROT	#l_cblas_#rot	S, D
#L_ROTM	#l_cblas_#rotm	S, D
#L_SWAP	#l_cblas_#swap	S, D, C, Z
#L_SCAL	#l_cblas_#scal	S, D, C, Z, CS, ZD
#L_COPY	#l_cblas_#copy	S, D, C, Z
#L_AXPY	#l_cblas_#axpy	S, D, C, Z
#L_DOT	#l_cblas_#dot	S, D, DS
#L_DOTU	#l_cblas_#dotu_sub	C, Z
#L_DOTC	#l_cblas_#dotc_sub	C, Z
#L_##DOT	#l_cblas_###dot	SDS
#L_NRM2, #L_#NRM2	#l_cblas_#nrm2, #l_cblas_##nrm2	S, D, SC, DZ
#L_ASUM, #L_#ASUM	#l_cblas_#asum, #l_cblas_##asum	S, D, SC, DZ
IL_#AMAX	#l_cblas_i#amax	S, D, C, Z

Table A.26 Level 2 sequential BLAS routines

Fortran BLAS sequential routine name	CBLAS sequential routine name	Supported precision (small letter for CBLAS)
#L_GEMV	#l_cblas_#gemv	S, D, C, Z
#L_GBMV	#l_cblas_#gbmv	S, D, C, Z
#L_HEMV	#l_cblas_#hemv	C, Z
#L_HBMV	#l_cblas_#hbmv	C, Z
#L_HPMV	#l_cblas_#hpmv	C, Z
#L_SYMV	#l_cblas_#symv	S, D
#L_SBMV	#l_cblas_#sbmv	S, D
#L_SPMV	#l_cblas_#spmv	S, D
#L_TRMV	#l_cblas_#trmv	S, D, C, Z
#L_TBMV	#l_cblas_#tbmv	S, D, C, Z
#L_TPMV	#l_cblas_#tpmv	S, D, C, Z
#L_TRSV	#l_cblas_#trsv	S, D, C, Z
#L_TBSV	#l_cblas_#tbsv	S, D, C, Z
#L_TPSV	#l_cblas_#tpsv	S, D, C, Z
#L_GER	#l_cblas_#ger	S, D
#L_GERU	#l_cblas_#geru	C, Z
#L_GERC	#l_cblas_#gerc	C, Z
#L_HER	#l_cblas_#her	C, Z
#L_HPR	#l_cblas_#hpr	C, Z
#L_HER2	#l_cblas_#her2	C, Z
#L_HPR2	#l_cblas_#hpr2	C, Z
#L_SYR	#l_cblas_#syr	S, D
#L_SPR	#l_cblas_#spr	S, D
#L_SYR2	#l_cblas_#syr2	S, D
#L_SPR2	#l_cblas_#spr2	S, D

Table A.27 Level 3 sequential BLAS routines

Fortran BLAS sequential routine name	CBLAS sequential routine name	Supported precision (small letter for CBLAS)
#L_GEMM	#l_cblas_#gemm	S, D, C, Z
#L_SYMM	#l_cblas_#symm	S, D, C, Z
#L_HEMM	#l_cblas_#hemm	C, Z
#L_SYRK	#l_cblas_#syrk	S, D, C, Z
#L_HERK	#l_cblas_#herk	C, Z
#L_SYR2K	#l_cblas_#syr2k	S, D, C, Z
#L_HER2K	#l_cblas_#her2k	C, Z
#L_TRMM	#l_cblas_#trmm	S, D, C, Z
#L_TRSM	#l_cblas_#trsm	S, D, C, Z

Table A.28 REAL*2 sequential BLAS routines

	Fortran BLAS sequential routine name	CBLAS sequential routine name
Level 1	SS_FJBLAS_SWAP_R16	ss_fjcblas_swap_r16
	SS_FJBLAS_SCAL_R16	ss_fjcblas_scal_r16
	SS_FJBLAS_COPY_R16	ss_fjcblas_copy_r16
	SS_FJBLAS_AXPY_R16	ss_fjcblas_axpy_r16
	SS_FJBLAS_DOT_R16	ss_fjcblas_dot_r16
	SS_FJBLAS_ASUM_R16	ss_fjcblas_asum_r16
	SS_FJBLAS_AMAX_I32_R16	ss_fjcblas_amax_i32_r16
Level 2	SS_FJBLAS_GEMV_R16	ss_fjcblas_gemv_r16
	SS_FJBLAS_GER_R16	ss_fjcblas_ger_r16
Level 3	SS_FJBLAS_GEMM_R16	ss_fjcblas_gemm_r16

Table A.29 Matrix copy and transpose sequential BLAS routines

Fortran BLAS sequential routine name	CBLAS sequential routine name
SS_FJBLAS_GEOCOPY_R16	ss_fjcblas_geocopy_r16
SS_FJBLAS_GEOCOPY_R32	ss_fjcblas_geocopy_r32
SS_FJBLAS_GEOCOPY_R64	ss_fjcblas_geocopy_r64
SS_FJBLAS_GEOCOPY_C32	ss_fjcblas_geocopy_c32
SS_FJBLAS_GEOCOPY_C64	ss_fjcblas_geocopy_c64

A.4.3 Matrix copy and transpose routines

This product supports routines to copy or transpose the matrix.

A.4.3.1 Fortran interface

Usage format

call fjblas_geocopy_r16(order, trans, rows, cols, alpha, a, lda, b, ldb)

call fjblas_geocopy_r32(order, trans, rows, cols, alpha, a, lda, b, ldb)

call fjblas_geocopy_r64(order, trans, rows, cols, alpha, a, lda, b, ldb)

call fjblas_geocopy_c32(order, trans, rows, cols, alpha, a, lda, b, ldb)

call fjblas_geocopy_c64(order, trans, rows, cols, alpha, a, lda, b, ldb)

Function

These routines perform copy, transpose, conjugate transpose or conjugate copy of matrix stored in two dimensional array. They can scale the matrix by a constant alpha.

B := alpha*op(A)

Input array and output array must not overlap.

The correspondence between routine names and alpha, a, and b types is as follows:

fjblas_geocopy_r16 : real(2)

fjblas_geocopy_r32 : real(4)

fjblas_geocopy_r64 : real(8)

fjblas_geocopy_c32 : complex(4)

fjblas_geocopy_c64 : complex(8)

Arguments

order	Input. character*1. Ordering of the matrix storage. order='R' or 'r' : row-major order='C' or 'c' : column-major Specify 'c' for a general Fortran two-dimensional array.
trans	Input. character*1. Trans specifies the operation to be performed. trans='N' or 'n' : Copy trans='T' or 't' : Transpose trans='R' or 'r' : Conjugate copy trans='C' or 'c' : Conjugate transpose
rows	Input. Integer. The number of rows of matrix b.
cols	Input. Integer. The number of columns of matrix b.
alpha	Input. The constant to multiply by the matrix.
a	Input. Two dimensional array a(lda,*). Source matrix
lda	Input. Integer. Leading dimension of a.
b	Output. Two dimensional array b(ldb,*). Destination matrix.
ldb	Input. Integer. Leading dimension of b.

How the matrix is copied or transposed from a to b by specifying order and trans.

When order = 'R' or 'r' and trans = 'N', 'n', 'R' or 'r',

Copy alpha * a (1: cols, 1: rows) and store it in b (1: cols, 1: rows).

lda > = cols, ldb > = cols

When order = 'R' or 'r' and trans = 'T', 't', 'C' or 'r',

Transpose alpha * a (1: cols, 1: rows) and store it in b (1: rows, 1: cols).

lda > = cols, ldb > = rows

When order = 'C' or 'c' and trans = 'N', 'n', 'R' or 'r',

Copy alpha * a (1: rows, 1: cols) and store it in b (1: rows, 1: cols).

lda > = rows, ldb > = rows

When order = 'C' or 'c' and trans = 'T', 't', 'C' or 'c',

Transpose alpha * a (1: rows, 1: cols) and store it in b (1: cols, 1: rows).

lda > = rows, ldb > = cols

A.4.3.2 C/C++ interface

Usage format

#include “cblas.h”

fjcblas_geocopy_r16(order, trans, rows, cols, alpha, (__fp16 *)a, lda, (__fp16 *)b, ldb)

fjcblas_geocopy_r32(order, trans, rows, cols, alpha, (float *)a, lda, (float *)b, ldb)

fjcblas_geocopy_r64(order, trans, rows, cols, alpha, (double *)a, lda, (double *)b, ldb)

fjcblas_geocopy_c32(order, trans, rows, cols, alpha, (fcomplex *)a, lda, (fcomplex *)b, ldb)

fjcblas_geocopy_c64(order, trans, rows, cols, alpha, (dcomplex *)a, lda, (dcomplex *)b, ldb)

fcomplex and dcomplex are single precision complex, double precision complex type defined in cblas.h.

Function

These routines perform copy, transpose, conjugate transpose or conjugate copy of matrix stored in two dimensional array. They also scale the matrix by a constant alpha.

B := alpha*op(A)

Input array and output array must not overlap.

The correspondence between routine names and alpha, a, and b types is as follows:

fjcblas_geocopy_r16 : __fp16

fjcblas_geocopy_r32 : float

fjcblas_geocopy_r64 : double

fjcblas_geocopy_c32 : fcomplex

fjcblas_geocopy_c64 : dcomplex

Arguments

order	Input. char. Ordering of the matrix storage. order=‘R’ or ‘r’ : row-major order=‘C’ or ‘c’ : column-major Specify 'c' for a general Fortran two-dimensional array.
trans	Input. char. Trans specifies the operation to be performed. trans=‘N’ or ‘n’ : Copy trans=‘T’ or ‘t’ : Transpose trans=‘R’ or ‘r’ : Conjugate copy trans=‘C’ or ‘c’ : Conjugate transpose
rows	Input. size_t. The number of rows of matrix b.

cols	Input. size_t. The number of columns of matrix b.
alpha	Input. The constant to multiply by the matrix.
a	Input. Two dimensional array a[][lda]. Source matrix
lda	Input. size_t. C fixed dimension of a.
b	Output. Two dimensional array b[][ldb]. Destination matrix.
ldb	Input. size_t. C fixed dimension of b.

How the matrix is copied or transposed from a to b by specifying order and trans.

When order = 'R' or 'r' and trans = 'N', 'n', 'R' or 'r',

Copy alpha * a[0…rows-1][0…cols-1] and store it in b[0…rows-1][0…cols-1].

lda > = cols, ldb > = cols

When order = 'R' or 'r' and trans = 'T', 't', 'C' or 'c',

Transpose alpha * a[0…rows-1][0…cols-1] and store it in b[0…cols-1][0…rows-1].

lda > = cols, ldb > = rows

When order = 'C' or 'c' and trans = 'N', 'n', 'R' or 'r',

Copy alpha * a[0…cols-1][0…rows-1] and store it in b[0…cols-1][0…colsrows-1].

lda > = rows, ldb > = rows

When order = 'C' or 'c' and trans = 'T', 't', 'C' or 'c',

Transpose alpha * a[0…cols-1][0…rows-1] and store it in b[0…rows-1][0…cols-1].

lda > = rows, ldb > = cols

Appendix B License

B.1 XBLAS

THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

B.2 LAPACK

Copyright (c) 1992-2011 The University of Tennessee and The University of Tennessee Research Foundation. All rights reserved.

Copyright (c) 2000-2011 The University of California Berkeley. All rights reserved.

Copyright (c) 2006-2012 The University of Colorado Denver. All rights reserved.

$COPYRIGHT$

Additional copyrights may follow

$HEADER$

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer listed in this license in the documentation and/or other materials provided with the distribution.

- Neither the name of the copyright holders nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

The copyright holders provide no reassurances that the source code provided does not infringe any patent, copyright, or any other intellectual property rights of third parties. The copyright holders disclaim any liability to any recipient for claims brought against

recipient by any third party for infringement of that parties
intellectual property rights.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

B.3 PLASMA

-- Innovative Computing Laboratory
-- Electrical Engineering and Computer Science Department
-- University of Tennessee
-- (C) Copyright 2008-2010

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

* Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
* Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
* Neither the name of the University of Tennessee, Knoxville nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS ``AS IS'' AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDERS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,

SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

B.4 ScaLAPACK

Copyright (c) 1992-2011 The University of Tennessee and The University
of Tennessee Research Foundation. All rights
reserved.
Copyright (c) 2000-2011 The University of California Berkeley. All
rights reserved.
Copyright (c) 2006-2011 The University of Colorado Denver. All rights
reserved.

$COPYRIGHT$

Additional copyrights may follow

$HEADER$

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer listed in this license in the documentation and/or other materials provided with the distribution.

- Neither the name of the copyright holders nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

The copyright holders provide no reassurances that the source code

provided does not infringe any patent, copyright, or any other intellectual property rights of third parties. The copyright holders disclaim any liability to any recipient for claims brought against recipient by any third party for infringement of that parties intellectual property rights.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.