BLAS LAPACK USER'S GUIDE

FUJITSU

Second Edition, January 2003
The contents of this manual may be revised without prior notice.

Preface

This manual describes the usage of the following two software.

BLAS thread-safe version V4.0
LAPACK thread-safe version V4.0

The products are the implementations of the public domain BLAS (Basic Linear Algebra Subprograms) and LAPACK (Linear Algebra PACKage), 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/) .

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 thread-safe subroutines

This section shows an example code that uses subroutines from BLAS and LAPACK thread-safe 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.

4 Compile and Link

The way of compiling the user's program containing calls to the products and of link-editing with the products is described.

5 Notes on Use

Several notes are provided on usage of the products.

Appendix A Routines List

The entire list of subprograms provided is given.

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

For a detailed specification of OpenMP Fortran please refer *OpenMP Fortran Application Program Interface Nov 2000 2.0* (http://www.openmp.org/).

Acknowledgement

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

Contents

List of Figures

List of Tables

1. Overview

BLAS and LAPACK thread-safe version are based on BLAS (Basic Linear Algebra Subprograms) and LAPACK (Linear Algebra PACKage). Each routine can be called from user programs written in Fortran with the CALL statement.

1.1 BLAS thread-safe version

BLAS is a library for vector and matrix operations. BLAS thread-safe version is based on BLAS provided on Netlib. BLAS includes 57 functions. The total number of routines for all precision types amounts to approximately 170.

BLAS thread-safe version 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

The thread-safe implementation of BLAS has exactly the same subroutine names and calling parameters as those of netlib baseline version. Only differences include

- the thread-safe version can be used in the environment of SMP (: Symmetric Multiple Processing) ,
- subroutines of the thread-safe version can be called from an OpenMP Fortran program, and
- the thread-safe version is to be link-edited with object programs created by Fujitsu OpenMP Fortran compiler.

The purpose of using BLAS thread-safe version is to have a subroutine concurrently perform operations on different sets of data that are independent from each other, and so reduce the turnaround time necessary to finish all the operations.

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 have to be available for reduction of elapsed time. The user will see the example code in the section of "3. Example program using thread-safe subroutines".

Note that BLAS thread-safe version is not a parallel library in which a subroutine is designed to solve a single problem using multiple threads, in other words, multiple CPUs. Instead, a subroutine of the thread-safe version is intended to solve independent multiple problems in parallel by multiple threads running on multiple CPUs. For example, the thread-safe subroutine for matrix multiplication can be called from multiple CPUs at a time, where one CPU executes one matrix multiplication.

Subroutines of BLAS thread-safe version also can be called from outside a parallel construct of OpenMP.

1.2 LAPACK thread-safe version

LAPACK is a library of linear algebra routines. LAPACK stread-safe version is based on LAPACK 3.0 provided on Netlib. LAPACK includes approximately 320 functions. The total number of routines for all precision types amounts to approximately 1300.

LAPACK provides the following routines.

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

LAPACK thread-safe version, a collection of subroutines for linear algebra is, like BLAS thread-safe version, called from 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.

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.

2.2 On-line Documentation

Several documentation of BLAS and LAPACK are available online from the domain http://www.netlib.org/. The following names are all as of January, 2003.

- LAPACK Users' Guide.

  The following URL provides an overview of calling forms for LAPACK routines, which includes the same information described in “Part 1 Guide” in the book of *LAPACK Users' Guide, Third Edition*.

  http://www.netlib.org/lapack/lug/lapack_lug.html

- Manual pages (i.e. “man” pages) for BLAS and LAPACK routines (a gzip tar file).

  http://www.netlib.org/lapack/manpages.tgz

3. Example program using thread-safe subroutines

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

3.1 Linear equations

3.1.1 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

3.1.2 Example code

In general, the user can choose the subroutine DGESV from LAPACK, a driver routine, to solve equations of the matrix form above. But for explanation purpose here, let's use kind of component routines DGETRF and DGETRS, which are called *computational* routines in the LAPACK book. The former is to compute an LU factorization of a given coefficient matrix A, and the latter to solve the factorized system for the solutions X.

Now let's assume we are going to solve the equations of order 200 with 80 different right hand side vectors. In the first place, we consider using conventional non thread-safe BLAS and LAPACK. An example program is shown below, where subroutine **inita** sets up the coefficient matrix in array **a**, the right hand side matrix consisting of b_i is set up using a given solutions for explanation purpose, and subroutine **check** checks the solutions x_i obtained back for correctness.

Example using conventional non thread-safe BLAS and LAPACK

```
      implicit real*8 (a-h,o-z)
      parameter(maxn=200,m=80,k=maxn+1)
      parameter(zero=0.0d0,one=1.0d0)
      real*8 a(k,maxn),aa(k,maxn),x(k,m),b(k,m)
      integer ip(maxn)
C     ============================================================
C     Define the matrix
C     ============================================================
      n=maxn
      call inita(a,k,n)
      do i=1,n
        do j=1,n
          aa(j,i)=a(j,i)
        end do
      end do
C     ============================================================
C     LU decomposition
C     ============================================================
      call dgetrf(n,n,a,k,ip,info)
```

```
C      ============================================================
C      Defeine the vectors
C      ============================================================
       do jm=1,m
         do jn=1,n
           x(jn,jm)=jn+jm
         end do
       end do
       call dgemm('N','N',n,m,n,one,aa,k,x,k,zero,b,k)
C      ============================================================
C      Solution
C      ============================================================
       call dgetrs('N',n,m,a,k,ip,b,k,info)
       if(info.ne.0) then
         write(6,*) 'error in dgetrs   info = ',info
         stop
       end if
C      ============================================================
C      Check result
C      ============================================================
       call check(a,b,k,n,m)
       end
```

Now let's use LAPACK thread-safe version. LU factorization is computed the same as the above example, but the procedure to obtain the solution *X* is a bit different and we are going to use multiple CPUs. In the next example subroutines from LAPACK thread-safe version is to be called. The 80 right hand side vectors are partitioned into groups, each having equal number, **mblk**, of right hand side vectors and those groups are passed at a time to subroutine DGETRS. Also note that initial setting up of right hand side *X* is done blockwise by calling thread-safe DGEMM for matrix multiplication.

Example BLAS, LAPACK thread-safe version

```
       implicit real*8 (a-h,o-z)
       parameter(maxn=200,m=80,mblk=4,k=maxn)
       parameter(zero=0.0d0,one=1.0d0)
       real*8 a(k,maxn),aa(k,maxn),x(k,m),b(k,m)
       integer ip(maxn)
C      ============================================================
C      Define the matrix
C      ============================================================
       n=maxn
       call inita(a,k,n)
       do i=1,n
         do j=1,n
           aa(j,i)=a(j,i)
         end do
       end do
C      ============================================================
C      LU decomposition
C      ============================================================
       call dgetrf(n,n,a,k,ip,info)
!$OMP PARALLEL PRIVATE(mb,info)
!$OMP DO
       do i=1,m,mblk
         mb=min(mblk,m-i+1)
C      ============================================================
C      Defeine the vectors
C      ============================================================
         do jm=1,mb
           do jn=1,n
             x(jn,jm+i-1)=jn+i+jm-1
           end do
         end do
         call dgemm('N','N',n,mb,n,one,aa,k,x(1,i),k,zero
     &             ,b(1,i),k)
C      ============================================================
C      Solution
C      ============================================================
         call dgetrs('N',n,mb,a,k,ip,b(1,i),k,info)
```

```
          if(info.ne.0) then
            write(6,*) 'error in dgetrs   info = ',info
            stop
          end if
        end do
!$OMP END PARALLEL
C       ============================================================
C       Check result
C       ============================================================
        call check(a,b,k,n,m)
        end
```

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 `-KOMP` options, 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.

4. Compile and Link

BLAS and LAPACK thread-safe versions are to be called from an OpenMP Fortran program. Compilation should be done using Fujitsu Fortran compiler. This section describes procedures of compilation through execution for these programs.

For users calling both BLAS and LAPACK routines, please follow information given in "4.2 LAPACK thread-safe version" and BLAS routines can be linked implicitly.

4.1 BLAS thread-safe version

BLAS can be used by linking it with user programs written in Fortran language. It is linked to the user program when `-lblasmt` or `-lblasmtp4` is specified on the `frt` command line. Two distinct version of library are provided, general version and Pentium®4 version which is tuned by using SSE2 instructions. In order to link with an appropriate version, specify the option `-lblasmt` or `-lblasmtp4` according to the hardware on which the program will be executed. Write `-lblasmt` or `-lblasmtp4` after the Fortran source and object names.

Other options relevant to optimization can be specified when needed.

Example 1:

Compile a user's program a.f, and link-edit it with the general version of BLAS library.

```
frt a.f –lblasmt
```

Example 2:

Compile a user's program a.f, and link-edit it with the Pentium®4 version of BLAS library.

```
frt a.f –lblasmtp4
```

When user program is written with OpenMP Fortran API, `-KOMP` option is required.

Example 3:

Compile a user's program a.f written in OpenMP Fortran and link-edit it with the Pentium®4 version of BLAS library.

```
frt –KOMP a.f –lblasmtp4
```

Execution of the program can be done by specifying its name after a prompt. The number of threads to be created can be specified by the environment variable OMP_NUM_THREADS. It is noted that when compiling with `-KOMP` option, all arrays used in the user's program are allocated in the stack area, so it might happen the stack area runs out of space. In that case, the user can expand the stack area by the command limit (in csh case) or ulimit (in sh or ksh cases).

4.2 LAPACK thread-safe version

LAPACK can be used by linking it with user programs written in Fortran language. It is linked to the user program when `-llapackmt` is specified on the `frt` command line. Since LAPACK calls BLAS, write `-lblasmt` or `-lblasmtp4` after `-llapackmt`.

Example 1:

Compile a user's program a.f, and link-edit it with LAPACK library and the general version BLAS library.

```
frt a.f -llapackmt –lblasmt
```

Example 2:

Compile a user's program a.f, and link-edit it with LAPACK library and the Pentium®4 version of BLAS library.

```
frt a.f  -llapackmt –lblasmtp4
```

When user program is written with OpenMP Fortran API, `-KOMP` option is required.

Example 3:

Compile a user's program a.f written in OpenMP Fortran and link-edit it with LAPACK library and the Pentium®4 version of BLAS library.

```
frt –KOMP a.f -llapackmt –lblasmtp4
```

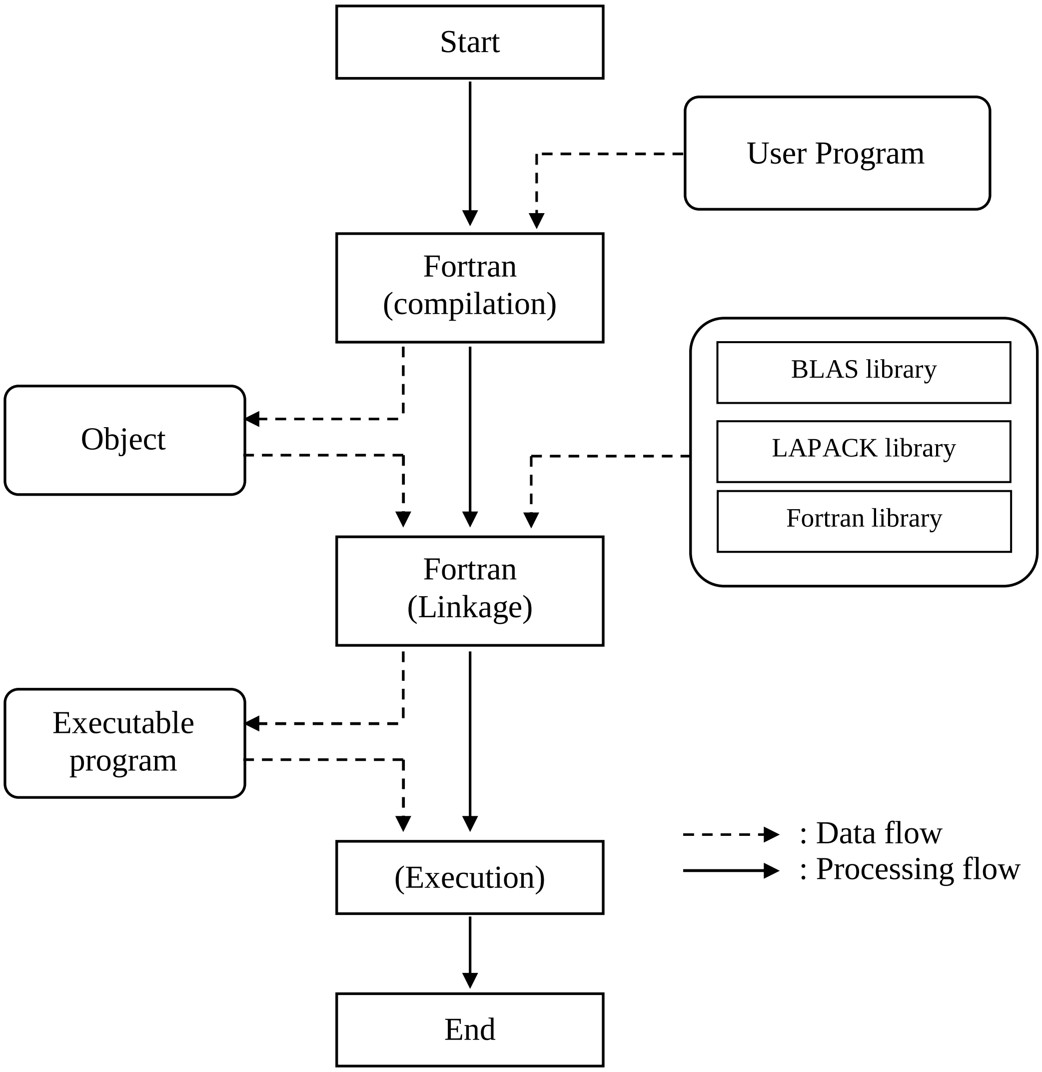

Figure 1 Flow of processing from compilation to execution (BLAS, LAPACK)

5. Notes on Use

Following are notes the user should be aware of for producing right results from routines.

5.1 Maximum number of threads

When calling BLAS or LAPACK, the maximum number of threads that can enter a subroutine at a time is 128.

5.2 Infinity and NaN

In the LAPACK version 3.0 from netlib, a set of subroutines have been added that expect infinities and NaN (not a number) defined in the IEEE standard, to be returned as results of zero-divide or overflow and not to terminate the computation.

Fujitsu Fortran fully conforms to the standard of such numbers. However, when the user specifies the option -NRtrap, error messages will come out. So, make sure to avoid using the option when compiling programs that call LAPACK routines.

5.3 Routines in the archive file

The library includes slave routines, the names of which starts #L_(# means S, D, C, S, I or X). The user needs to be careful not to duplicate subroutine names with them.

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 thread-safe version

The routines that are supplied with BLAS are listed in Table A.1 to Table A.4. The slave routines included in this software are listed in Table A.5

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.

Table A.1 Level 1 BLAS routines

Routine name	Supported precision
#ROTG	S, D
#ROTMG	S, D
#ROT	S, D
#ROTM	S, D
#SWAP	S, D, C, Z
#SCAL	S, D, C, Z, CS, ZD
#COPY	S, D, C, Z
#AXPY	S, D, C, Z
#DOT	S, D, DS
#DOTU	C, Z
#DOTC	C, Z
##DOT	SDS
#NRM2	S, D, SC, DZ
#ASUM	S, D, SC, DZ
I#AMAX	S, D, C, Z

Table A.2 Level 2 BLAS routines

Routine name	Supported precision
#GEMV	S, D, C, Z
#GBMV	S, D, C, Z
#HEMV	C, Z
#HBMV	C, Z
#HPMV	C, Z
#SYMV	S, D
#SBMV	S, D
#SPMV	S, D
#TRMV	S, D, C, Z
#TBMV	S, D, C, Z
#TPMV	S, D, C, Z
#TRSV	S, D, C, Z
#TBSV	S, D, C, Z
#TPSV	S, D, C, Z
#GER	S, D
#GERU	C, Z
#GERC	C, Z
#HER	C, Z
#HPR	C, Z
#HER2	C, Z
#HPR2	C, Z
#SYR	S, D
#SPR	S, D
#SYR2	S, D
#SPR2	S, D

Table A.3 Level 3 BLAS routines

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

Table A.4 Sparse BLAS

Routine name	Supported precision
#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 Slave routines for BLAS

Routine name	Contents
DCABS1, LSAME, XREBLA	Included opened BLAS

A.2 LAPACK thread-safe version

The routines that are supplied with LAPACK are listed in Table A.6 to Table A.7.

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 is unique and do not depend on data type.

The symbol * shows new routines in version 3.0 of LAPACK on Netlib.

Table A.6 Driver and Computational routines of LAPACK

Real routine	Complex routine
SBDSDC*	—
SBDSQR	CBDSQR
SDISNA	—
SGBBRD	CGBBRD
SGBCON	CGBCON
SGBEQU	CGBEQU
SGBRFS	CGBRFS
SGBSV	CGBSV
SGBSVX	CGBSVX
SGBTRF	CGBTRF
SGBTRS	CGBTRS
SGEBAK	CGEBAK
SGEBAL	CGEBAL

Real routine	Complex routine
SGEBRD	CGEBRD
SGECON	CGECON
SGEEQU	CGEEQU
SGEES	CGEES
SGEESX	CGEESX
SGEEV	CGEEV
SGEEVX	CGEEVX
SGEHRD	CGEHRD
SGELQF	CGELQF
SGELS	CGELS
SGELSD *	CGELSD *
SGELSS	CGELSS
SGELSY *	CGELSY *

Table A.6 Driver and Computational routines of LAPACK (continued)

Real routine	Complex routine	Real routine	Complex routine
SGEQLF	CGEQLF	SORMBR	CUNMBR
SGEQP3 *	CGEQP3 *	SORMHR	CUNMHR
SGEQRF	CGEQRF	SORMLQ	CUNMLQ
SGERFS	CGERFS	SORMQL	CUNMQL
SGERQF	CGERQF	SORMQR	CUNMQR
SGESDD *	CGESDD *	SORMRQ	CUNMRQ
SGESV	CGESV	SORMRZ *	CUNMRZ *
SGESVD	CGESVD	SORMTR	CUNMTR
SGESVX	CGESVX	SPBCON	CPBCON
SGETRF	CGETRF	SPBEQU	CPBEQU
SGETRI	CGETRI	SPBRFS	CPBRFS
SGETRS	CGETRS	SPBSTF	CPBSTF
SGGBAK	CGGBAK	SPBSV	CPBSV
SGGBAL	CGGBAL	SPBSVX	CPBSVX
SGGES *	CGGES *	SPBTRF	CPBTRF
SGGESX *	CGGESX *	SPBTRS	CPBTRS
SGGEV *	CGGEV *	SPOCON	CPOCON
SGGEVX *	CGGEVX *	SPOEQU	CPOEQU
SGGHRD	CGGHRD	SPORFS	CPORFS
SGGGLM	CGGGLM	SPOSV	CPOSV
SGGLSE	CGGLSE	SPOSVX	CPOSVX
SGGQRF	CGGQRF	SPOTRF	CPOTRF
SGGRQF	CGGRQF	SPOTRI	CPOTRI
SGGSVD	CGGSVD	SPOTRS	CPOTRS
SGGSVP	CGGSVP	SPPCON	CPPCON
SGTCON	CGTCON	SPPEQU	CPPEQU
SGTRFS	CGTRFS	SPPRFS	CPPRFS
SGTSV	CGTSV	SPPSV	CPPSV
SGTSVX	CGTSVX	SPPSVX	CPPSVX
SGTTRF	CGTTRF	SPPTRF	CPPTRF
SGTTRS	CGTTRS	SPPTRI	CPPTRI
SHGEQZ	CHGEQZ	SPPTRS	CPPTRS
SHSEIN	CHSEIN	SPTCON	CPTCON
SHSEQR	CHSEQR	SPTEQR	CPTEQR
SOPGTR	CUPGTR	SPTRFS	CPTRFS
SOPMTR	CUPMTR	SPTSV	CPTSV
SORGBR	CUNGBR	SPTSVX	CPTSVX
SORGHR	CUNGHR	SPTTRF	CPTTRF
SORGLQ	CUNGLQ	SPTTRS	CPTTRS
SORGQL	CUNGQL	SSBEV	CHBEV
SORGQR	CUNGQR	SSBEVD	CHBEVD
SORGRQ	CUNGRQ	SSBEVX	CHBEVX
SORGTR	CUNGTR	SSBGST	CHBGST

Table A.6 Driver and Computational routines of LAPACK (continued)

Real routine	Complex routine	Real routine	Complex routine
SSBGV	CHBGV	SSYEVX	CHEEVX
SSBGVD *	CHBGVD *	SSYGST	CHEGST
SSBGVX *	CHBGVX *	SSYGV	CHEGV
SSBTRD	CHBTRD	SSYGVD *	CHEGVD *
SSPCON	CSPCON	SSYGVX *	CHEGVX *
—	CHPCON	SSYRFS	CSYRFS
SSPEV	CHPEV	—	CHERFS
SSPEVD	CHPEVD	SSYSV	CSYSV
SSPEVX	CHPEVX	—	CHESV
SSPGST	CHPGST	SSYSVX	CSYSVX
SSPGV	CHPGV	—	CHESVX
SSPGVD *	CHPGVD *	SSYTRD	CHETRD
SSPGVX *	CHPGVX *	SSYTRF	CSYTRF
SSPRFS	CSPRFS	—	CHETRF
—	CHPRFS	SSYTRI	CSYTRI
SSPSV	CSPSV	—	CHETRI
—	CHPSV	SSYTRS	CSYTRS
SSPSVX	CSPSVX	—	CHETRS
—	CHPSVX	STBCON	CTBCON
SSPTRD	CHPTRD	STBRFS	CTBRFS
SSPTRF	CSPTRF	STBTRS	CTBTRS
—	CHPTRF	STGEVC	CTGEVC
SSPTRI	CSPTRI	STGEXC *	CTGEXC *
—	CHPTRI	STGSEN *	CTGSEN *
SSPTRS	CSPTRS	STGSJA	CTGSJA
—	CHPTRS	STGSNA *	CTGSNA *
SSTEBZ	—	STGSYL *	CTGSYL *
SSTEDC	CSTEDC	STPCON	CTPCON
SSTEGR *	CSTEGR *	STPRFS	CTPRFS
SSTEIN	CSTEIN	STPTRI	CTPTRI
SSTEQR	CSTEQR	STPTRS	CTPTRS
SSTERF	—	STRCON	CTRCON
SSTEV	—	STREVC	CTREVC
SSTEVD	—	STREXC	CTREXC
SSTEVR *	—	STRRFS	CTRRFS
SSTEVX	—	STRSEN	CTRSEN
SSYCON	CSYCON	STRSNA	CTRSNA
—	CHECON	STRSYL	CTRSYL
SSYEV	CHEEV	STRTRI	CTRTRI
SSYEVD	CHEEVD	STRTRS	CTRTRS
SSYEVR *	CHEEVR *	STZRZF	CTZRZF

Table A.7 Auxiliary routines of LAPACK

Real routine	Complex routine
—	CLACGV
—	CLACRM
—	CLACRT
—	CLAESY
—	CROT
—	CSPMV
—	CSPR
—	CSROT
—	CSYMV
—	CSYR
—	ICMAX1
ILAENV	—
LSAME	—
LSAMEN	—
—	SCSUM1
SGBTF2	CGBTF2
SGEBD2	CGEBD2
SGEHD2	CGEHD2
SGELQ2	CGELQ2
SGEQL2	CGEQL2
SGEQR2	CGEQR2
SGERQ2	CGERQ2
SGESC2 *	CGESC2 *
SGETC2 *	CGETC2 *
SGETF2	CGETF2
SGTTS2 *	CGTTS2 *
SLABAD	—
SLABRD	CLABRD
SLACON	CLACON
SLACPY	CLACPY
SLADIV	CLADIV
SLAE2	—
SLAEBZ	—
SLAED0	CLAED0
SLAED1	—
SLAED2	—
SLAED3	—
SLAED4	—
SLAED5	—
SLAED6	—
SLAED7	CLAED7
SLAED8	CLAED8

Real routine	Complex routine
SLAED9	—
SLAEDA	—
SLAEIN	CLAEIN
SLAEV2	CLAEV2
SLAEXC	—
SLAG2	—
SLAGS2 *	—
SLAGTF	—
SLAGTM	CLAGTM
SLAGTS	—
SLAGV2 *	—
SLAHQR	CLAHQR
SLAHRD	CLAHRD
SLAIC1	CLAIC1
SLALN2	—
SLALS0 *	CLALS0 *
SLALSA *	CLALSA *
SLALSD *	CLALSD *
SLAMCH	—
SLAMRG	—
SLANGB	CLANGB
SLANGE	CLANGE
SLANGT	CLANGT
SLANHS	CLANHS
SLANSB	CLANSB
—	CLANHB
SLANSP	CLANSP
—	CLANHP
SLANST	CLANHT
SLANSY	CLANSY
—	CLANHE
SLANTB	CLANTB
SLANTP	CLANTP
SLANTR	CLANTR
SLANV2	—
SLAPLL	CLAPLL
SLAPMT	CLAPMT
SLAPY2	—
SLAPY3	—
SLAQGB	CLAQGB
SLAQGE	CLAQGE
SLAQP2 *	CLAQP2 *

Table A.7 Auxiliary routines of LAPACK (continued)

Real routine	Complex routine
SLAQPS *	CLAQPS *
SLAQSB	CLAQSB
SLAQSP	CLAQSP
SLAQSY	CLAQSY
SLAQTR	—
SLAR1V *	CLAR1V *
SLAR2V	CLAR2V
SLARF	CLARF
SLARFB	CLARFB
SLARFG	CLARFG
SLARFT	CLARFT
SLARFX	CLARFX
SLARGV	CLARGV
SLARNV	CLARNV
SLARRB *	—
SLARRE *	—
SLARRF *	—
SLARRV *	CLARRV
SLARTG	CLARTG
SLARTV	CLARTV
SLARUV	—
SLARZ *	CLARZ *
SLARZB *	CLARZB *
SLARZT *	CLARZT *
SLAS2	—
SLASCL	CLASCL
SLASD0 *	—
SLASD1 *	—
SLASD2 *	—
SLASD3 *	—
SLASD4 *	—
SLASD5 *	—
SLASD6 *	—
SLASD7 *	—
SLASD8 *	—
SLASD9 *	—
SLASDA *	—
SLASDQ *	—
SLASDT *	—
SLASET	CLASET
SLASQ1	—
SLASQ2	—

Real routine	Complex routine
SLASQ3	—
SLASQ4	—
SLASQ5 *	—
SLASQ6 *	—
SLASR	CLASR
SLASRT	—
SLASSQ	CLASSQ
SLASV2	—
SLASWP	CLASWP
SLASY2	—
SLASYF	CLASYF
—	CLAHEF
SLATBS	CLATBS
SLATDF *	CLATDF *
SLATPS	CLATPS
SLATRD	CLATRD
SLATRS	CLATRS
SLATRZ *	CLATRZ *
SLAUU2	CLAUU2
SLAUUM	CLAUUM
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
SSYGS2	CHEGS2
SSYTD2	CHETD2
SSYTF2	CSYTF2
—	CHETF2
STGEX2 *	CTGEX2 *
STGSY2 *	CTGSY2 *
STRTI2	CTRTI2
XERBLA	—