High Performance Linear Algebra Package LAPACK90

Jerzy Waśniewski[*] Jack Dongarra[†]

March 28, 1998

Abstract

LAPACK90 is a set of FORTRAN90 subroutines which interfaces FORTRAN90 with LAPACK.

All LAPACK driver subroutines (including expert drivers) and some LAPACK computationals have both generic LAPACK90 interfaces and generic LAPACK77 interfaces. The remaining computationals have only generic LAPACK77 interfaces. In both types of interfaces, no distinction is made between single and double precision or between real and complex data types.

1 Introduction

The high performance linear algebra package, LAPACK is adapted for the new FORTRAN standard, FORTRAN 90/95. For convenience, we use the name LAPACK 77 to denote the existing FORTRAN 77 LAPACK package, and LAPACK 90 to denote the new FORTRAN 90 interface which is describe here.

We provide background information and references for LAPACK, ScaLAPACK, FORTRAN 90 and HPF in this section. The end of this section contains very brief statements of LAPACK90 as well.

1.1 LAPACK

LAPACK is a library of FORTRAN 77 subroutines for solving the most commonly occurring problems in numerical linear algebra. It has been designed to be efficient on a wide range of modern, high-performance computers. The name LAPACK is an acronym for Linear Algebra PACKage.

[*]The Danish Computing Centre for Research and Education (UNI•C), Technical University of Denmark, Building 304, DK-2800 Lyngby, Denmark, Email: jerzy.wasniewski@uni-c.dk

[†]Department of Computer Science, University of Tennessee, 107 Ayres Hall, Knoxville, TN 37996-1301, USA and Mathematical Sciences Section, Oak Ridge National Laboratory, P.O.Box 2008, Bldg. 6012, Oak Ridge, TN 37831-6367, USA; email: dongarra@cs.utk.edu

LAPACK provides routines for solving systems of simultaneous linear equations, least-squares solutions of linear systems of equations, eigenvalue problems, and singular value problems. The associated matrix factorizations (LU, Cholesky, QR, SVD, Schur, generalized Schur) are also provided, as are related computations such as reordering of the Schur factorizations and estimating condition numbers. Dense and banded matrices are handled, but not general sparse matrices. In all areas, similar functionality is provided for real and complex matrices, in both single and double precision.

The original goal of the LAPACK project was to make the widely used EISPACK and LINPACK libraries run efficiently on shared-memory vector and parallel processors. On these machines, LINPACK and EISPACK are inefficient because their memory access patterns disregard the multi-layered memory hierarchies of the machines, thereby spending too much time moving data instead of doing useful floating-point operations. LAPACK addresses this problem by reorganizing the algorithms to use block matrix operations, such as matrix multiplication, in the innermost loops. These block operations can be optimized for each architecture to account for the memory hierarchy, and so provide a transportable way to achieve high efficiency on diverse modern machines. LAPACK requires that highly optimized block matrix operations be already implemented on each machine.

LAPACK routines are written so that as much of the computation as possible is performed by calls to the Basic Linear Algebra Subprograms[13] (BLAS). While LINPACK and EISPACK are based on the vector operation kernels of the Level 1 BLAS. LAPACK is designed to exploit the Level 3 BLAS – a set of specifications for FORTRAN subprograms that do various types of matrix multiplication and the solution of triangular systems with multiple right-hand sides. Because of the coarse granularity of the Level 3 BLAS operations, their use promotes high efficiency on many high-performance computers, particularly if specially coded implementations are provided by the manufacturer.

Highly efficient, machine-specific implementations of the BLAS are available for many modern high-performance computers. The BLAS enable LAPACK routines to achieve high performance with transportable software. A model FORTRAN implementation of the BLAS is available from netlib[5] in the BLAS library. It is not expected to perform as well as a specially tuned implementation on most high-performance computers. On some machines, it may give much worse performance. But it allows users to run LAPACK software on machines that do not offer any other implementation of the BLAS.

For more information on LAPACK and references on BLAS, LINPACK and EISPACK see [13, 1].

1.2 ScaLAPACK

ScaLAPACK is a library of high-performance, linear algebra routines for distributed memory message-passing MIMD (Multiple Instruction Multiple Data) computers and networks of workstations supporting PVM[7] (Parallel Virtual Machine) and/or MPI[12] (Message Passing Interface). ScaLAPACK is a continuation of the LAPACK project (see section 1.1). The name ScaLAPACK is an acronym for Scalable Linear Algebra PACKage, or Scalable LAPACK. Both libraries (LAPACK and ScaLAPACK) contain routines for solving systems of linear equations, least squares problems, and eigenvalue problems. The goals of both projects are efficiency (to run as fast as possible), scalability (as the problem size and number of processors grow), reliability (including error bounds), portability (across all important parallel machines), flexibility (so users can construct new routines from well-designed parts), and ease of use (by making the interface to LAPACK and ScaLAPACK look as similar as possible). Many of these goals, particularly portability, are aided by developing and promoting standards , especially for low-level communication and computation routines. ScaLAPACK has been successful in attaining these goals, limiting most machine dependencies to two standard libraries called the BLAS (Basic Linear Algebra Subprograms) and BLACS[14] (Basic Linear Algebra Communication Subprograms). LAPACK runs on any machine where the BLAS[13] are available, and ScaLAPACK runs on any machine where both the BLAS and the BLACS are available.

The library is currently written in FORTRAN 77 (with the exception of a few symmetric eigenproblem auxiliary routines written in C to exploit IEEE arithmetic) in a Single Program Multiple Data (SPMD) style using explicit message passing for interprocessor communication.

For more information on ScaLAPACK and references on BLAS, BLACS, PBLAS, PVM and MPI see [13, 14, 6, 2, 7, 12].

1.3 FORTRAN 90

FORTRAN has always been the principal computer language used in the fields of science, engineering, and numerical computing. A series of revisions to the standard defining successive versions of the language has progressively enhanced its power and kept it competitive with several generations of rivals. The present FORTRAN standard is 90/95. Below is a summary of the new features:

- Array operations.
- Pointers.
- Improved facilities for numerical computations including a set of numerical inquiry functions.
- Parameterization of the intrinsic types, to permit processors to support short integers, very large character sets, more than two precisions for real and complex, and packed logicals.

- User-defined derived data types composed of arbitrary data structures and operations upon those structures.
- Facilities for defining collections called "modules", useful for global data definitions and for procedure libraries. These support a safe method of encapsulating derived data types.
- Requirements on a compiler to detect the use of constructs that do not conform to syntax of the language or are obsolescent.
- A few source form, more appropriate to use at a terminal
- New control constructs such as the SELECT CASE construct and a new form of the DO.
- The ability to write internal procedures and recursive procedures, and to call procedures with optional and keyword arguments.
- Dynamic storage (automatic arrays, allocatable arrays, and pointers).
- Improvements to the input-output facilities, including handling partial records and a standardized NAMELIST facility.
- Many new intrinsic procedures.

Combined, the new features contained in FORTRAN 90/95 ensure that the FORTRAN language will continue to be used successfully in the future. The fact that it contains the whole of FORTRAN 77 as a subset means that conversion to FORTRAN 90/95 is as simple as conversion to another FORTRAN 77 processor. For more information on FORTRAN 90/95 see [10].

1.4 High Performance FORTRAN (HPF)

FORTRAN is reaching its limitations on the latest generations of high performance computers. FORTRAN was originally developed for serial machines with linear memory architectures. In the past several years, it has become increasingly apparent that a language design relying on this architectural feature creates difficulties when executing on parallel machines. One symptom of this is the proliferation of parallel FORTRAN dialects, each specialized to the machine where it was first implemented. As the number of competing parallel machines on the market increases, the lack of a standard parallel FORTRAN is becoming increasingly problematic. HPF solves this problem. The overriding goal of HPF was therefore to produce a dialect of FORTRAN that could be used on variety of parallel machines. HPF is an extension of FORTRAN 90/95. The array calculation and dynamic storage allocation features of FORTRAN 90, and the **FORALL** statement, the **PURE** and **EXTRINSIC** attributes of FORTRAN 95, make it natural base for HPF. The new HPF language futures fall into four categories with respect to FORTRAN 90/95:

- New directives.
- New language syntax.
- Library routines.
- Language restrictions.

For more information on HPF see [8].

1.5 LAPACK for FORTRAN 90

All LAPACK driver subroutines (including expert drivers) and some LAPACK computationals have both generic LAPACK90 interfaces and generic LAPACK77 interfaces. The remaining computationals have only generic LAPACK77 interfaces. In both types of interfaces, no distinction is made between single and double precision or between real and complex data types. The use of the LAPACK90 (LAPACK77) interface requires the user to specify the F90_LAPACK (F77_LAPACK) module.

For example, the GESV driver subroutine, which solves a general system of linear equations, can be called in the following ways:

- CALL LA_GESV(A, B, IPIV=ipiv, INFO=info)
  or
- CALL LA_GESV(N, NRHS, A, LDA, IPIV, B, LDB, INFO)

The module F90_LAPACK is needed in the first case in which the LAPACK90 interface package is called. The module F77_LAPACK is needed in the second case in which the LAPACK77 package is directly called.

The present implementation of the LAPACK90 can be summarized in the following titles:

- Driver Routines for Linear Equations.
- Expert Driver Routines for Linear Equations.
- Driver Routines for Linear Least Squares Problems.
- Driver Routines for generalized Linear Least Squares Problems.
- Driver Routines for Standard Eigenvalue and Singular Value Problems.
- Divide and Conquer Driver Routines for Standard Eigenvalue Problems.
- Expert Driver Routines for Standard Eigenvalue Problems.
- Driver Routines for Generalized Eigenvalue and Singular Value Problems.

- Some Computational Routines for Linear Equations and Eigenproblems.

The LAPACK90 library is successively updated and it is available from netlib (see [5, 4]).

1.6 ScaLAPACK for HPF

The HPF ScaLAPACK interface project started in several places (see [9, 11]) including UNI•C. The work at UNI•C is not described yet. The report is in preparation. Several ScaLAPACK subroutines and test programs are interfaced with HPF.

2 Interface Blocks for LAPACK 77

```
PROGRAM EXAMPLE
  USE LA_PRECISION, ONLY: WP => SP
  USE F77_LAPACK, ONLY: LA_GESV
  IMPLICIT NONE
  CHARACTER(LEN=*), PARAMETER :: FMT = '(7(1X,F9.3))'
  INTEGER :: J, INFO, N, NRHS, LDA, LDB
  INTEGER, ALLOCATABLE :: IPIV(:)
  REAL(WP), ALLOCATABLE :: A(:,:), B(:,:)
  N = 5; NRHS = 2
  ALLOCATE( A(N,N), B(N,NRHS), IPIV(N) )
  CALL RANDOM_NUMBER(A)
  DO J = 1, NRHS; B(:,J) = SUM( A, DIM=2)*J; ENDDO
  LDA = N; LDB = N
  CALL LA_GESV( N, NRHS, A, LDA, IPIV, B, LDB, INFO )
  WRITE(*,*) 'INFO = ', INFO
  IF( NRHS < 6 .AND. N < 11 )THEN
    WRITE(*,*) 'The solution:'
    DO J = 1, NRHS; WRITE (*,FMT) B(:,J); ENDDO
  ENDIF
END PROGRAM EXAMPLE
```

Figure 1: Example1: Module F77_LAPACK is used.

All LAPACK77 driver subroutines (including expert drivers) and LAPACK77 computationals have generic interfaces. No distinction is made between single and double precision or between real and complex data types. The use of the LAPACK77 generic interface requires the user to specify the F77_LAPACK module.

Example 1 in fig. 1 demonstrates the use of a LAPACK77 generic interface. The program solves a linear system of equations $AX = B$, where A is a square matrix and B and X are rectangular matrices.

Remarks:

- **Statement 2** includes SP interface block from the LA_PRECISION module. WP will be used internally as SP. The interface block SP defines the precision (see page 21), in this case single precision. The program works in double precision if DP replaces SP.
- **Statement 3** includes the LA_GESV interface block from F77_LAPACK module.
- **Statement 8.** REAL(WP) defines variables A and B, in this case allocatable arrays A and B in single precision. The program will work in complex if COMPLEX replaces REAL.
- **Statement 14.** The generic interface name LA_GESV is replaced during the compilation phase by the proper interface body (see page 15). In this case SGESV replaces LA_GESV.

Appendix A contains, as examples, the generic interfaces of LA_GETRF and LA_GESV for LAPACK77. The generic interfaces of the LAPACK77 driver and computational routines determine the F77_LAPACK module.

For more information see references [3, 4].

3 Interface Blocks for LAPACK 90

All LAPACK90 driver subroutines (including expert drivers) and some LAPACK90 computationals have generic interfaces. No distinction is made between single and double precision or between real and complex data types. The use of the LAPACK90 generic interface requires the user to specify the F90_LAPACK module.

Example 2 in fig. 2 demonstrates the use of a LAPACK90 generic interface. The program solves a linear system of equations $AX = B$, where A is a square matrix and B and X are rectangular matrices. The computation in example 2 is the same as that in example 1. However the program is shorter and the call of LA_GESV is simpler.

Remarks:

- **Statement 2** includes SP interface block from the LA_PRECISION module. WP is internally used as SP. The interface block SP defines the precision (see page 21), in this case single precision. The program works in double precision if DP replaces SP.

```
 1  PROGRAM EXAMPLE
 2    USE LA_PRECISION, ONLY: WP => SP
 3    USE f90_LAPACK, ONLY: LA_GESV
 4    IMPLICIT NONE
 5    CHARACTER(LEN=*), PARAMETER :: FMT = '(7(1X,F9.3))'
 6    INTEGER :: J, N, NRHS
 7    REAL(WP), ALLOCATABLE :: A(:,:), B(:,:)
 8    N = 5; NRHS = 2
 9    ALLOCATE( A(N,N), B(N,NRHS) )
10    CALL RANDOM_NUMBER(A)
11    DO J = 1, NRHS; B(:,J) = SUM( A, DIM=2)*J; ENDDO
12    CALL LA_GESV( A, B )
13    IF( NRHS < 6 .AND. N < 11 )THEN
14      WRITE(*,*) 'The solution:'
15      DO J = 1, NRHS; WRITE (*,FMT) B(:,J); ENDDO
16    ENDIF
17  END PROGRAM EXAMPLE
```

Figure 2: Example2: Module F90_LAPACK is used.

- **Statement 3** includes the LA_GESV interface block from F90_LAPACK module.
- **Statement 7.** REAL(WP) defines variables A and B, in this case allocatable arrays A and B in single precision. The program works in complex if COMPLEX replaces REAL.
- **Statement 12.** The generic interface name LA_GESV is replaced during the compilation phase by the proper interface body (see page 18). In this case SGESV_F90 replaces LA_GESV because of SP and REAL and because the shape of array B is (:,:). LA_GESV is replaced by SGESV1_F90 if the array B has shape (:).

Example 3 in fig. 3 demonstrates the use of both LAPACK77 and LAPACK90 generic interfaces. The program also solves a linear system of equations $AX = B$, where A is a square matrix, and B and X are rectangular matrices.

Appendix B contains, as examples, the generic interfaces of LA_GETRF and LA_GESV for LAPACK90. The generic interfaces of the LAPACK90 driver and computational routines determine the F90_LAPACK module.

For more information see references [3, 4].

```
PROGRAM EXAMPLE
  USE LA_PRECISION, ONLY: WP => SP
  USE f77_LAPACK, ONLY: F77GESV => LA_GESV
  USE f90_LAPACK, ONLY: F90GESV => LA_GESV
  IMPLICIT NONE
  INTEGER :: INFO, J, LDA, LDB, N, NRHS
  INTEGER, ALLOCATABLE :: IPIV(:)
  REAL :: T0, T1, T2
  REAL(WP), ALLOCATABLE :: A(:,:), B(:,:)
  N = 500; NRHS = 2
  ALLOCATE( A(N,N), B(N,NRHS), IPIV(N) )
  CALL RANDOM_NUMBER(A)
  DO J = 1, NRHS; B(:,J) = SUM( A, DIM=2)*J; ENDDO
  LDA = N; LDB = N
  CALL CPU_TIME(T0); CALL CPU_TIME(T1); T0 = T1-T0
  CALL F77GESV( N, NRHS, A, LDA, IPIV, B, LDB, INFO )
  CALL CPU_TIME(T2)
  WRITE(*,*) 'INFO and CPUTIME of F77GESV ', INFO, T2-T1-T0
  CALL CPU_TIME(T1); CALL F90GESV( A, B ); CALL CPU_TIME(T2)
  WRITE(*,*) 'CPUTIME of F90GESV ', T2-T1-T0
END PROGRAM EXAMPLE
```

Figure 3: Example3: Both modules F77_LAPACK and F90_LAPACK are used.

4 Code of LAPACK90 Routines

Two LAPACK90 interface routines, LA_GESV and LA_GETRI are listed in the appendix C. The code of such routine can be divided in the following parts:

- Heading of the routine
  - Subroutine or function statement
  - USE statements
    - LA_PRECISION module
    - LA_AUXMOD (auxiliary) module if needed
    - F77_LAPACK module
  - IMPLICIT NONE statement
  - Argument specifications
- Argument descriptions (comments)
- Local variable declaration

- Executable statements
  - Local variables initialization
  - Testing the arguments
  - Work space allocation if needed
  - Writing warning message if needed
  - Calling the LAPACK77 routine
  - Work space deallocation if needed
  - Calling the error trapping routine (see page 22)
- end of routine statement

The routines LA_GESV (page 19) and LA_GETRI (page 19) illustrate the above.

The LA_PRECISION module and the ERINFO subroutine are illustrated in appendix D.

5 LAPACK90 Documentation

The LAPACK90 documentation can be divided into three categories.

Routine text. Every LAPACK90 interface routine contains documentation as comments, including the purpose, argument specification, argument description, and further details if necessary.

On-line documentation. The documentation of the LAPACK90 library is available on the Web at address "http://www.netlib.org/lapack90/". It gives very brief information but there are links to more detailed information if needed. First is given general LAPACK90 information. If you need LAPACK77 information you can click on "LAPACK Users' Guide". If you want to down load the LAPACK90 installation package you should click on "lapack90/lapack90.tar.gz". If you need LAPACK90 specific information you should click on "LAPACK90 homepage".

In the "LAPACK90 homepage" you will find a brief description of every LAPACK90 interface subroutine. For example,

- CALL LA_GESV (A, B, IPIV=ipiv, INFO=info)
  Solves a general system of linear equations AX = B.

For more information click on "LA_GESV".

LAPACK90 Users' Guide. The guide gives some theoretical background information and describes every user-callable subroutine. Purposes of the subroutines, argument specifications, argument descriptions, and examples are provided. The documentation of the LA_GESV subroutine is listed in appendix E.

The manual is also applicable to the LAPACK FORTRAN90 and ScaLAPACK HPF interfaces.

A CD ROM with examples from the book are attached to the Users' Guide.

6 LAPACK90 Test Programs

The LAPACK90 test programs can be divided into three categories.

1. Every LAPACK90 interface program has a test program. These programs were used by the authors in developing the LAPACK90 interface. The programs test the interface routines, the computation, and the error exits. These programs can be used as examples for LAPACK90 beginners. The programs are collected in the directory LAPACK90/EXAMPLES.

2. Some of the LAPACK77 test programs were adapted for LAPACK90.

3. A new series of easy-to-use test programs are under development. The user can run such a program, interpret the results, and examine the numerical accuracy. The tests are already developed for the driver routines of the section on linear system of equations. The results of the test program for LA_GESV are listed in appendix F. These tests will be distributed with the LAPACK90 package.

7 LAPACK90 User Callable Routines

Appendix G contains a short description of all LAPACK90 routines. The call of the routine and a brief statement of its purpose are given. For example, for LA_GESV:

- CALL LA_GESV(A, B, IPIV=ipiv, INFO=info)
  Solves a general system of linear equations AX = B.

Arguments A and B must always be specified while $IPIV$ and $INFO$ are optional. For more routine descriptions see appendix G.

Acknowledgments

This research was partially supported by the Danish Natural Science Research Council through a grant for the EPOS (Efficient Parallel Algorithms for Optimization and Simulation) project and in part by Oak Ridge National Laboratory, managed by Lockheed Martin Energy Research Corp. for the U.S. Department of Energy under contract number DE-AC05-96OR22464.

References

[1] E. Anderson, Z. Bai, C. H. Bischof, J. Demmel, J. J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, S. Ostrouchov and D. C. Sorensen. *LAPACK Users' Guide Release 2.0.* SIAM, Philadelphia, 1995.

[2] L.S. Blackford, J. Choi, A. Ceary, E. D'Azevedo, J. Demmel, I. Dhilon, J. Dongarra, S. Hammarling, G. Henry, A. Petitet, K. Stanley, D. Walker, and R.C. Whaley. *ScaLAPACK Users' Guide.* SIAM, Philadelphia, 1997.

[3] L.S Blackford, J.J. Dongarra, J. Du Croz, S. Hammarling, and J. Waśniewski. *LAPACK Working Note 117, A Proposal for a FORTRAN 90 Interface for LAPACK.* Report UNIC-96-10, UNI•C, Lyngby, Denmark, 1995. Report ut-cs-96-341, University of Tennessee, Computer Science Department, Knoxville, July, 1995.

[4] L.S. Blackford, J.J. Dongarra, J. Du Croz, S. Hammarling, and J. Waśniewski. *LAPACK90 - FORTRAN90 version of LAPACK.* On web: http://webhotel.uni-c.dk/para/lapack90/ and http://www.netlib.org/lapack90/ (1997)

[5] S. Browne, J. Dongarra, E. Grosse, and T. Rowan. *The Netlib Mathematical Software Repository.* D-Lib Magazine, Sep, 1995, Accessible at http://www.dlib.org/

[6] J. Choi, J. Dongarra, S. Ostrouchov, A. Petitet, D. Walker, and, R. C. Whaley. *A Proposal for a Set of Parallel Basic Linear Algebra Subprograms.* Universeety of Tennessee at Knoxville, Technical Report, CS-95-292, May 1995. Accessible at http://www.netlib.org/lapack/lawns/index.html (lapack/lawns/lawn100.ps).

[7] A. Geist, A. Beguelin, J. Dongarra, W. Jiang, R. Manchek, and V. Sunderam *PVM: A Users' Guide and Tutorial for Networked Parallel Computing.* MIT Press, 1994.

[8] C.H. Koelbel, D.B. Lovemann, R.S. Schreiber, G.L. Steele Jr., and M.E. Zosel. *The High Performance FORTRAN Handbook.* The MIT Press Cambridge, Massachusetts, London, England, 1994.

[9] P.A.R. Lorenzo, A. Müller, Y. Murakami, and B.J.N. Wylie. High Performance FORTRAN Interfacing to ScaLAPACK. In J. Waśniewski, J. Dongarra, K. Madsen, and D. Olesen (Eds.), Applied Parallel Computing,

Industrial Computation and Optimization, Third International Workshop, PARA'96, Lyngby, Denmark, August 1996, Proceedings, Lecture Notes in Computer Science No. 1184, Springer-Verlag, 1996, pp. 457-466

[10] M. Metcalf and J. Reid. *FORTRAN 90 Explained.* Oxford, New York, Tokyo, Oxford University Press, 1990.

[11] *R.C. Whaley. HPF Interface to ScaLAPACK.* On web: http://www.netlib.org/scalapack/prototype/ (1997).

[12] M. Snir, S. Otto, S. Huss-Lederman, D. Walker, and J. Dongarra. *MPI: The Complete Reference.* The MIT Press Cambridge, Massachusetts, 1996.

[13] *BLAS (Basic Linear Algebra Subprograms).* See at netlib http://www.netlib.org/blas/index.html

[14] *BLACS (BLAS (Basic Linear Algebra Communication Subprograms).* See at netlib http://www.cs.utk.edu/ rwhaley/Blacs.html

A LAPACK77 Generic Interface Blocks

LA_GETRF

```
MODULE F77_LAPACK

  INTERFACE LA_GETRF

    SUBROUTINE SGETRF( M, N, A, LDA, PIV, INFO )
      USE LA_PRECISION, ONLY: WP => SP
      INTEGER, INTENT(IN) :: LDA, M, N
      INTEGER, INTENT(OUT) :: INFO
      INTEGER, INTENT( OUT ) :: PIV( * )
      REAL(WP), INTENT( INOUT ) :: A( LDA, * )
    END SUBROUTINE SGETRF

    SUBROUTINE DGETRF( M, N, A, LDA, PIV, INFO )
      USE LA_PRECISION, ONLY: WP => DP
      INTEGER, INTENT(IN) :: LDA, M, N
      INTEGER, INTENT(OUT) :: INFO
      INTEGER, INTENT( OUT ) :: PIV( * )
      REAL(WP), INTENT( INOUT ) :: A( LDA, * )
    END SUBROUTINE DGETRF

    SUBROUTINE CGETRF( M, N, A, LDA, PIV, INFO )
      USE LA_PRECISION, ONLY: WP => SP
      INTEGER, INTENT(IN) :: LDA, M, N
      INTEGER, INTENT(OUT) :: INFO
      INTEGER, INTENT( OUT ) :: PIV( * )
      COMPLEX(WP), INTENT( INOUT ) :: A( LDA, * )
    END SUBROUTINE CGETRF

    SUBROUTINE ZGETRF( M, N, A, LDA, PIV, INFO )
      USE LA_PRECISION, ONLY: WP => DP
      INTEGER, INTENT(IN) :: LDA, M, N
      INTEGER, INTENT(OUT) :: INFO
      INTEGER, INTENT( OUT ) :: PIV( * )
      COMPLEX(WP), INTENT( INOUT ) :: A( LDA, * )
    END SUBROUTINE ZGETRF

  END INTERFACE

END MODULE F77_LAPACK
```

LA_GESV

```
MODULE F77_LAPACK

  INTERFACE LA_GESV

    SUBROUTINE SGESV( N, NRHS, A, LDA, PIV, B, LDB, INFO )
      USE LA_PRECISION, ONLY: WP => SP
      INTEGER, INTENT(IN) :: LDA, LDB, NRHS, N
      INTEGER, INTENT(OUT) :: INFO
      INTEGER, INTENT(OUT) :: PIV(*)
      REAL(WP), INTENT(INOUT) :: A(LDA,*), B(LDB,*)
    END SUBROUTINE SGESV

    SUBROUTINE DGESV( N, NRHS, A, LDA, PIV, B, LDB, INFO )
      USE LA_PRECISION, ONLY: WP => DP
      INTEGER, INTENT(IN) :: LDA, LDB, NRHS, N
      INTEGER, INTENT(OUT) :: INFO
      INTEGER, INTENT(OUT) :: PIV(*)
      REAL(WP), INTENT(INOUT) :: A(LDA,*), B(LDB,*)
    END SUBROUTINE DGESV

    SUBROUTINE CGESV( N, NRHS, A, LDA, PIV, B, LDB, INFO )
      USE LA_PRECISION, ONLY: WP => SP
      INTEGER, INTENT(IN) :: LDA, LDB, NRHS, N
      INTEGER, INTENT(OUT) :: INFO
      INTEGER, INTENT(OUT) :: PIV(*)
      COMPLEX(WP), INTENT(INOUT) :: A(LDA,*), B(LDB,*)
    END SUBROUTINE CGESV

    SUBROUTINE ZGESV( N, NRHS, A, LDA, PIV, B, LDB, INFO )
      USE LA_PRECISION, ONLY: WP => DP
      INTEGER, INTENT(IN) :: LDA, LDB, NRHS, N
      INTEGER, INTENT(OUT) :: INFO
      INTEGER, INTENT(OUT) :: PIV(*)
      COMPLEX(WP), INTENT(INOUT) :: A(LDA,*), B(LDB,*)
    END SUBROUTINE ZGESV

    MODULE PROCEDURE SGESV1, DGESV1, CGESV1, ZGESV1

  END INTERFACE
```

LA_GESV (cont)

```
CONTAINS
  SUBROUTINE SGESV1( N, NRHS, A, LDA, PIV, B, LDB, INFO )
    USE LA_PRECISION, ONLY: WP => SP
    INTEGER, INTENT(IN) :: LDA, LDB, NRHS, N
    INTEGER, INTENT(OUT) :: INFO
    INTEGER, INTENT(OUT) :: PIV(*)
    REAL(WP), INTENT(INOUT) :: A(LDA,*), B(*)
    INTERFACE
      SUBROUTINE SGESV( N, NRHS, A, LDA, PIV, B, LDB, INFO )
        USE LA_PRECISION, ONLY: WP => SP
        INTEGER, INTENT(IN) :: LDA, LDB, NRHS, N
        INTEGER, INTENT(OUT) :: INFO
        INTEGER, INTENT(OUT) :: PIV(*)
        REAL(WP), INTENT(INOUT) :: A(LDA,*), B(LDB,*)
      END SUBROUTINE SGESV
    END INTERFACE
    CALL SGESV( N, NRHS, A, LDA, PIV, B, LDB, INFO )
  END SUBROUTINE SGESV1
  SUBROUTINE DGESV1( N, NRHS, A, LDA, PIV, B, LDB, INFO )
    USE LA_PRECISION, ONLY: WP => DP

             ...            ...            ...

    CALL DGESV( N, NRHS, A, LDA, PIV, B, LDB, INFO )
  END SUBROUTINE DGESV1
  SUBROUTINE CGESV1( N, NRHS, A, LDA, PIV, B, LDB, INFO )
    USE LA_PRECISION, ONLY: WP => SP

             ...            ...            ...

    CALL CGESV( N, NRHS, A, LDA, PIV, B, LDB, INFO )
  END SUBROUTINE CGESV1
  SUBROUTINE ZGESV1( N, NRHS, A, LDA, PIV, B, LDB, INFO )
    USE LA_PRECISION, ONLY: WP => DP

             ...            ...            ...

    CALL ZGESV( N, NRHS, A, LDA, PIV, B, LDB, INFO )
  END SUBROUTINE ZGESV1
END MODULE F77_LAPACK
```

B LAPACK90 Generic Interface Blocks

LA_GETRF

```
MODULE F90_LAPACK
  INTERFACE LA_GETRF
    SUBROUTINE SGETRF_F90( A, IPIV, RCOND, NORM, INFO )
      USE LA_PRECISION, ONLY: WP => SP
      CHARACTER(LEN=1), INTENT(IN), OPTIONAL :: NORM
      INTEGER, INTENT(OUT), OPTIONAL :: INFO
      REAL(WP), INTENT( OUT ), OPTIONAL :: RCOND
      INTEGER, INTENT( OUT ), OPTIONAL :: IPIV( : )
      REAL(WP), INTENT( INOUT ) :: A( :, : )
    END SUBROUTINE SGETRF_F90
    SUBROUTINE DGETRF_F90( A, IPIV, RCOND, NORM, INFO )
      USE LA_PRECISION, ONLY: WP => DP
      CHARACTER(LEN=1), INTENT(IN), OPTIONAL :: NORM
      INTEGER, INTENT(OUT), OPTIONAL :: INFO
      REAL(WP), INTENT( OUT ), OPTIONAL :: RCOND
      INTEGER, INTENT( OUT ), OPTIONAL :: IPIV( : )
      REAL(WP), INTENT( INOUT ) :: A( :, : )
     END SUBROUTINE DGETRF_F90
    SUBROUTINE CGETRF_F90( A, IPIV, RCOND, NORM, INFO )
      USE LA_PRECISION, ONLY: WP => SP

             ...            ...            ...

    END SUBROUTINE CGETRF_F90
    SUBROUTINE ZGETRF_F90( A, IPIV, RCOND, NORM, INFO )
      USE LA_PRECISION, ONLY: WP => DP
      CHARACTER(LEN=1), INTENT(IN), OPTIONAL :: NORM
      INTEGER, INTENT(OUT), OPTIONAL :: INFO
      REAL(WP), INTENT( OUT ), OPTIONAL :: RCOND
      INTEGER, INTENT( OUT ), OPTIONAL :: IPIV( : )
      COMPLEX(WP), INTENT( INOUT ) :: A( :, : )
    END SUBROUTINE ZGETRF_F90
  END INTERFACE
END MODULE F90_LAPACK
```

LA_GESV

```
MODULE F90_LAPACK
  INTERFACE LA_GESV
    SUBROUTINE SGESV_F90( A, B, IPIV, INFO )
      USE LA_PRECISION, ONLY: WP => SP
      INTEGER, INTENT(OUT), OPTIONAL :: INFO
      INTEGER, INTENT(OUT), OPTIONAL :: IPIV(:)
      REAL(WP), INTENT(INOUT) :: A(:,:), B(:,:)
    END SUBROUTINE SGESV_F90
    SUBROUTINE SGESV1_F90( A, B, IPIV, INFO )
      USE LA_PRECISION, ONLY: WP => SP
      INTEGER, INTENT(OUT), OPTIONAL :: INFO
      INTEGER, INTENT(OUT), OPTIONAL :: IPIV(:)
      REAL(WP), INTENT(INOUT) :: A(:,:), B(:)
    END SUBROUTINE SGESV1_F90
    SUBROUTINE DGESV_F90( A, B, IPIV, INFO )
      USE LA_PRECISION, ONLY: WP => DP

             ...            ...            ...

    END SUBROUTINE DGESV_F90
    SUBROUTINE DGESV1_F90( A, B, IPIV, INFO )
      USE LA_PRECISION, ONLY: WP => DP

             ...            ...            ...

    END SUBROUTINE CGESV_F90
    SUBROUTINE CGESV1_F90( A, B, IPIV, INFO )
      USE LA_PRECISION, ONLY: WP => SP

             ...            ...            ...

    END SUBROUTINE CGESV1_F90
    SUBROUTINE ZGESV_F90( A, B, IPIV, INFO )
      USE LA_PRECISION, ONLY: WP => DP

             ...            ...            ...

    END SUBROUTINE ZGESV1_F90
  END INTERFACE
END MODULE F90_LAPACK
```

C LA_GESV and LA_GETRI subroutines

LA_GESV

```
 SUBROUTINE SGESV_F90( A, B, IPIV, INFO )
 USE LA_PRECISION, ONLY: WP => SP
 USE LA_AUXMOD, ONLY: ERINFO
 USE F77_LAPACK, ONLY: GESV_F77 => LA_GESV
 IMPLICIT NONE
 INTEGER, INTENT(OUT), OPTIONAL :: INFO
 INTEGER, INTENT(OUT), OPTIONAL, TARGET :: IPIV(:)
 REAL(WP), INTENT(INOUT) :: A(:,:), B(:,:)
!-------------------------------------------------------------
              (Argument descriptions)
!-------------------------------------------------------------
 CHARACTER(LEN=7), PARAMETER :: SRNAME = 'LA_GESV'
 INTEGER :: LINFO, ISTAT, ISTAT1, SIPIV, N, NRHS, LDA, LDB
 INTEGER, POINTER :: LPIV(:)
 INTRINSIC SIZE, PRESENT, MAX
!-------------------------------------------------------------
 LINFO = 0; ISTAT = 0; N = SIZE(A,1); NRHS = SIZE(B,1)
 IF( PRESENT(IPIV) )THEN; SIPIV = SIZE(IPIV)
 ELSE; SIPIV = N; ENDIF
 IF( N < 0 .OR. SIZE(A,2) /= N )THEN; LINFO = -1
 ELSE IF( SIZE( B, 1 ) /= N .OR. NRHS < 0 )THEN; LINFO = -2
 ELSE IF( SIPIV /= SIZE(A,1) )THEN; LINFO = -3
 ELSE IF( N > 0 )THEN
   IF( PRESENT(IPIV) )THEN; LPIV => IPIV
   ELSE; ALLOCATE( LPIV(SIZE(A,1)), STAT = ISTAT ); END IF
   IF( ISTAT == 0 ) THEN; LDA = MAX(1,N); LDB = MAX(1,N)
     CALL GESV_F77( N, NRHS, A, LDA, LPIV, B, LDB, LINFO )
   ELSE; LINFO = -100; END IF
   IF( .NOT.PRESENT(IPIV) ) DEALLOCATE(LPIV,STAT = ISTAT1)
 END IF
 CALL ERINFO( LINFO, SRNAME, INFO, ISTAT )
 END SUBROUTINE SGESV_F90
```

LA_GETRI

```
SUBROUTINE SGETRI_F90( A, IPIV, INFO )
  USE LA_PRECISION, ONLY: WP => SP
  USE LA_AUXMOD, ONLY: ERINFO
  USE F77_LAPACK, ONLY: GETRI_F77 => LA_GETRI, &
                        ILAENV_F77 => ILAENV
  IMPLICIT NONE
  INTEGER, INTENT(OUT), OPTIONAL :: INFO
  INTEGER, INTENT(IN) :: IPIV(:)
  REAL(WP), INTENT(INOUT) :: A(:,:)
               (Argument Descriptions)
   CHARACTER(LEN=8), PARAMETER :: SRNAME = 'LA_GETRI'
   CHARACTER(LEN=6), PARAMETER :: BSNAME = 'SGETRI'
   INTEGER    :: LINFO, N, LD, LWORK, ISTAT, ISTAT1, NB
   REAL(WP), POINTER :: WORK(:)
   INTRINSIC SIZE, MAX
!----------------------------------------------------------
   N = SIZE(A,1); LINFO = 0; LD = MAX(1,N); ISTAT = 0
   IF( SIZE( A, 2 ) /= N .OR. N < 0 )THEN; LINFO = -1
   ELSE IF( SIZE( IPIV ) /= N )THEN; LINFO = -2
   ELSE IF( N > 0 )THEN
     NB = ILAENV_F77( 1, BSNAME, ' ', N, -1, -1, -1 )
     IF( NB < 1 .OR. NB >= N )THEN; NB = 1; END IF
     LWORK = MAX( N*NB, 1 )
     ALLOCATE(WORK(LWORK), STAT=ISTAT)
     IF( ISTAT /= 0 )THEN; DEALLOCATE(WORK, STAT=ISTAT1)
       LWORK = MAX(1,N); ALLOCATE(WORK(LWORK), STAT=ISTAT)
       IF( ISTAT == 0 ) CALL ERINFO( -200, SRNAME, LINFO )
     END IF
     IF( LINFO == 0 )THEN
       CALL GETRI_F77( N, A, LD, IPIV, WORK, LWORK, LINFO )
     ELSE; LINFO = -100; END IF
     DEALLOCATE(WORK, STAT=ISTAT1)
   END IF
   CALL ERINFO(LINFO,SRNAME,INFO,ISTAT)
END SUBROUTINE SGETRI_F90
```

D Auxiliary Routines

LA_PRECISION

```
MODULE LA_PRECISION
  INTEGER, PARAMETER :: SP=KIND(1.0), DP=KIND(1.0D0)
END MODULE LA_PRECISION
```

LA_AUXMOD

```
MODULE LA_AUXMOD
  INTERFACE
    SUBROUTINE ERINFO(LINFO, SRNAME, INFO, ISTAT)
      CHARACTER( LEN = * ), INTENT(IN) :: SRNAME
      INTEGER, INTENT(IN) :: LINFO
      INTEGER, INTENT(OUT), OPTIONAL :: INFO
      INTEGER, INTENT(IN), OPTIONAL :: ISTAT
    END SUBROUTINE ERINFO
    INTEGER FUNCTION LA_WS_GELS( VER, M, N, NRHS, TRANS )
      CHARACTER( LEN=1 ), INTENT(IN) :: TRANS, VER
      INTEGER, INTENT(IN) :: M, N, NRHS
    END FUNCTION LA_WS_GELS
    INTEGER FUNCTION LA_WS_GELSS( VER, M, N, NRHS )
      CHARACTER(LEN=1), INTENT(IN) :: VER
      INTEGER, INTENT(IN) :: M, N, NRHS
    END FUNCTION LA_WS_GELSS
  END INTERFACE
  CONTAINS
  LOGICAL FUNCTION LSAME( CA, CB )
    CHARACTER(LEN=1), INTENT(IN) :: CA, CB
! LSAME  TESTS IF CA IS THE SAME LETTER AS CB REGARDLESS OF CASE.

             ...            ...            ...

  END FUNCTION LSAME
END MODULE LA_AUXMOD
```

LA_ERINFO

```
SUBROUTINE ERINFO(LINFO, SRNAME, INFO, ISTAT)
  IMPLICIT NONE
  CHARACTER( LEN = * ), INTENT(IN) :: SRNAME
  INTEGER, INTENT(IN) :: LINFO
  INTEGER, INTENT(OUT), OPTIONAL :: INFO
  INTEGER, INTENT(IN), OPTIONAL :: ISTAT

  IF( ( ( LINFO < 0 .AND. LINFO > -200 ) .OR. LINFO > 0 ) &
                    .AND. .NOT.PRESENT(INFO) )THEN
    WRITE (*,*) 'Terminated in LAPACK\_90 subroutine ', &
                SRNAME
    WRITE (*,*) 'Error indicator, INFO = ',LINFO
    IF( PRESENT(ISTAT) )THEN; IF( ISTAT /= 0 ) THEN
      IF( LINFO == -100 )THEN
        WRITE (*,*) 'ALLOCATE causes STATUS = ', ISTAT
      ELSE
        WRITE (*,*) 'LINFO = ', LINFO, ' not expected'
      END IF
    END IF; END IF
    STOP
    ELSE IF( LINFO <= -200 ) THEN
      WRITE(*,*) '++++++++++++++++++++++++++++++++++++++++++++'
      WRITE(*,*) '*** WARNING, INFO = ', LINFO, ' WARNING ***'
      IF( LINFO == -200 )THEN

             ...            ...            ...

      WRITE(*,*) '++++++++++++++++++++++++++++++++++++++++++++'
    END IF
      IF( PRESENT(INFO) ) INFO = LINFO
  END IF
END SUBROUTINE ERINFO
```

E Documentation of LA_GESV

Purpose

LA_GESV computes the solution to a real or complex system of linear equations $AX = B$, where A is a square matrix and B and X are rectangular matrices or vectors. Gaussian elimination with row interchanges is used to factor A as $A = P^T LU$, where P is a permutation matrix, L is unit lower triangular, and U is upper triangular. The factored form of A is then used to solve the system of equations $AX = B$.

Specification

SUBROUTINE LA_GESV(A, B, IPIV=ipiv, INFO=info)
    type(*wp*), INTENT(INOUT) :: A(:,:), *rhs*
    INTEGER, INTENT(OUT), OPTIONAL :: IPIV(:)
    INTEGER, INTENT(OUT), OPTIONAL :: INFO
    where
    type ::= REAL | COMPLEX
    wp ::= KIND(1.0) | KIND(1.0D0)
    rhs ::= B(:,:) | B(:)

Arguments

A – (*input/output*) **REAL** or **COMPLEX** square array, shape $(:,:)$.

- On entry, the matrix A.
- On exit, the factors L and U from the factorization $A = P^T LU$; the unit diagonal elements of L are not stored.

B – (*input/output*) **REAL** or **COMPLEX** array, shape $(:,:)$ or $(:)$, and $size(\mathbf{B}, 1) = size(\mathbf{A}, 1)$ or $size(\mathbf{B}) = size(\mathbf{A}, 1)$.

- On entry, the right-hand side vector(s) of matrix B in the system of equations $AX = B$.
- On exit, if there is no error, the matrix of solution vector(s) X.

IPIV – *Optional* (*output*) **INTEGER** array, shape $(:)$, $size(\mathbf{IPIV}) = size(\mathbf{A}, 1)$.

- The indices that define the permutation matrix P; row i of the matrix was interchanged with row $\mathbf{IPIV}_i$.

INFO – *Optional* (*output*) **INTEGER**.

- $= 0$: successful exit.
  < 0 : if **INFO** $= -i$, the i^{th} argument has an illegal value.
  > 0 : if **INFO** $= i$, then $U_{i,i} = 0$. A is singular and no solution was computed.

If **INFO** is not present and an error occurs, then the program is terminated with an error message.

Examples

The results below are computed with $\epsilon = 1.1921_{10} - 07$.

Example 1 (from Program LA_GESV_EXAMPLE)

$$A = \begin{pmatrix} 0 & 2 & 3 & 5 & 4 \\ 1 & 0 & 5 & 6 & 6 \\ 7 & 6 & 8 & 0 & 5 \\ 4 & 6 & 0 & 3 & 9 \\ 5 & 9 & 0 & 0 & 8 \end{pmatrix}, \qquad B = \begin{pmatrix} 14 & 28 & 42 \\ 18 & 36 & 54 \\ 26 & 52 & 78 \\ 22 & 44 & 66 \\ 22 & 44 & 66 \end{pmatrix}$$

Arrays **A** and **B** on entry:

A

0	2	3	5	4
1	0	5	6	6
7	6	8	0	5
4	6	0	3	9
5	9	0	0	8

B

14	28	42
18	36	54
26	52	78
22	44	66
22	44	66

The call:

CALL LA_GESV(A, B)

B on exit:

B

1.0000000	2.0000000	3.0000012
1.0000000	2.0000000	3.0000000
1.0000000	2.0000000	2.9999993
1.0000001	2.0000002	3.0000012
1.0000000	2.0000000	2.9999990

The solution of the system $AX = B$ is:

$$X = \begin{pmatrix} 1.0000000 & 2.0000000 & 3.0000012 \\ 1.0000000 & 2.0000000 & 3.0000000 \\ 1.0000000 & 2.0000000 & 2.9999993 \\ 1.0000001 & 2.0000002 & 3.0000012 \\ 1.0000000 & 2.0000000 & 2.9999990 \end{pmatrix}.$$

Example 2 (from Program LA_GESV_EXAMPLE)

A on entry: As in Example 1.
B on entry: $B_{:,1}$, where B is the input matrix in Example 1.

The call:

CALL LA_GESV(A, B(:,1), IPIV, INFO)

A, **B**(:, **1**), **IPIV** and **INFO** on exit:

A

7.0000000	6.0000000	8.0000000	0.0000000	5.0000000
0.7142857	4.7142859	−5.7142859	0.0000000	4.4285712
0.0000000	0.4242424	5.4242425	5.0000000	2.1212122
0.5714286	0.5454544	−0.2681566	4.3407826	4.2960901
0.1428571	−0.1818182	0.5195531	0.7837837	1.6216215

B(:, **1**)	**IPIV**
1.0000000	3
1.0000000	5
1.0000000	3
1.0000001	4
1.0000000	5

$INFO = 0$

Matrices L, U and P:

$$L = \begin{pmatrix} 1.0000000 & & & & \\ 0.7142857 & 1.0000000 & & & \\ 0.0000000 & 0.4242424 & 1.0000000 & & \\ 0.5714286 & 0.5454544 & -0.2681566 & 1.0000000 & \\ 0.1428571 & -0.1818182 & 0.5195531 & 0.7837837 & 1.0000000 \end{pmatrix}$$

$$U = \begin{pmatrix} 7.0000000 & 6.0000000 & 8.0000000 & 0.0000000 & 5.0000000 \\ & 4.7142859 & -5.7142859 & 0.0000000 & 4.4285712 \\ & & 5.4242425 & 5.0000000 & 2.1212122 \\ & & & 4.3407826 & 4.2960901 \\ & & & & 1.6216215 \end{pmatrix}$$

$$P = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{pmatrix}$$

The solution of the system $AX = b$ is:

$$x = \begin{pmatrix} 1.0000000 \\ 1.0000000 \\ 1.0000000 \\ 1.0000001 \\ 1.0000000 \end{pmatrix}.$$

F The LA_GESV test results

Test Runs Correctly

```
SGESV Test Example Program Results.
LA\_GESV LAPACK subroutine solves a dense general
linear system of equations, Ax = b.
Threshold value of test ratio = 10.00 the machine eps = 0.11921E-06
```

```
--------------------------------------------------------
3 matrices were tested with 4 tests. NRHS was 50 and one.
The biggest tested matrix was 300 x 300
12 tests passed.
0 tests failed.
--------------------------------------------------------
9 error exits tests were ran
9 tests passed.
0 tests failed.
```

Test Partly Fails

```
SGESV Test Example Program Results.
LA\_GESV LAPACK subroutine solves a dense general
linear system of equations, Ax = b.
Threshold value of test ratio = 5.00 the machine eps = 0.11921E-06
--------------------------------------------------------
Test 1 -- 'CALL LA\_GESV( A, B, IPIV, INFO )', Failed.
Matrix 300 x 300 with 50 rhs.
INFO =  0
|| A ||1 = 14.4323969  COND = 2.0686414E+02
|| X ||1 = 2.2516827E+05  || B - AX ||1 = 2.0583858
ratio = || B - AX || / ( || A ||*|| X ||*eps ) = 5.3133821
--------------------------------------------------------
3 matrices were tested with 4 tests. NRHS was 50 and one.
The biggest tested matrix was 300 x 300
11 tests passed.
1 test failed.
--------------------------------------------------------
9 error exits tests were ran
9 tests passed.
0 tests failed.
```

G LAPACK90 User Callable Routines

Driver Routines for Linear Equations

- CALL LA_GESV(A, B, IPIV=ipiv, INFO=info)
  Solves a general system of linear equations AX = B.

- CALL LA_GBSV(AB, B, KL=kl, IPIV=ipiv, INFO=info)
  Solves a general band system of linear equations AX = B.

- CALL LA_GTSV(DL, D, DU, B, INFO=info)
  Solves a general tridiagonal system of linear equations AX = B.

- CALL LA_POSV(A, B, UPLO=uplo, INFO=info)
  Solves a symmetric/Hermitian positive definite system of linear equations AX = B.

- CALL LA_PPSV(AP, B, UPLO=uplo, INFO=info)
  Solves a symmetric/Hermitian positive definite (packed storage) system of linear equations AX = B.

- CALL LA_PBSV(AB, B, UPLO=uplo, INFO=info)
  Solves a symmetric/Hermitian positive definite band system of linear equations AX = B.

- CALL LA_PTSV(D, E, B, INFO=info)
  Solves a symmetric/Hermitian positive definite tridiagonal system of linear equations AX = B.

- CALL LA_SYSV / LA_HESV(A, B, UPLO=uplo, IPIV=ipiv, &
  INFO=info)
  Solves a symmetric/Hermitian/complex indefinite system of linear equations AX = B.

- CALL LA_SPSV /LA_HPSV(AP, B, UPLO=uplo, IPIV=ipiv, &
  INFO=info)
  Solves a symmetric/Hermitian/complex indefinite (packed storage) system of linear equations AX = B.

Expert Driver Routines for Linear Equations

- CALL LA_GESVX(A, B, X, AF=af, IPIV=ipiv, FACT=fact, &
  TRANS=trans, EQUED=equed, R=r, C=c, &
  FERR=ferr, BERR=berr, RCOND=rcond, &
  RPVGRW=rpvgrw, INFO=info)
  Solves a general system of linear equations AX = B. Error bounds on the solution and a condition estimate are also provided.

- CALL LA_GBSVX(AB, B, X, KL=kl, ABF=abf, IPIV=ipiv, &
  FACT=fact, TRANS=trans, EQUED=equed, &
  R=r, C=c, FERR=ferr, BERR=berr, &
  RCOND=rcond, RPVGRW=rpvgrv, INFO=info)
  Solves a general band system of linear equations AX = B. Error bounds on the solution and a condition estimate are also provided.

-
  ```
  CALL LA_GTSVX( DL, D, DU, B, X=x, DLF=dlf, DF=df, &
                 DUF=duf, DU2=du2, IPIV=ipiv, FACT=fact, &
                 TRANS=trans, FERR=ferr, BERR=berr, &
                 RCOND=rcond, INFO=info)
  ```
  Solves a general tridiagonal system of linear equations $AX = B$. Error bounds on the solution and a condition estimate are also provided.

-
  ```
  CALL LA_POSVX( A, B, X, UPLO=uplo, AF=af, FACT=fact, &
                 EQUED=equed, S=s, FERR=ferr, &
                 BERR=berr, RCOND=rcond, INFO=info )
  ```
  Solves a symmetric/Hermitian positive definite system of linear equations $AX = B$. Error bounds on the solution and a condition estimate are also provided.

-
  ```
  CALL LA_PPSVX( AP, B, X, UPLO=uplo, AFP=afp, FACT=fact, &
                 EQUED=equed, S=s, FERR=ferr, &
                 BERR=berr, RCOND=rcond, INFO=info )
  ```
  Solves a symmetric/Hermitian positive definite (packed storage) system of linear equations $AX = B$. Error bounds on the solution and a condition estimate are also provided.

-
  ```
  CALL LA_PBSVX(AB, B, X, UPLO=uplo, AFB=afb, FACT=fact, &
                 EQUED=equed, S=s, FERR=ferr, &
                 BERR=berr, RCOND=rcond, INFO=info )
  ```
  Solves a symmetric/Hermitian positive definite band system of linear equations $AX = B$. Error bounds on the solution and a condition estimate are also provided.

-
  ```
  CALL LA_PTSVX( D, E, B, X, DF=df, EF=ef, FACT=fact, &
                 FERR=ferr, BERR=berr, RCOND=rcond, &
                 INFO=info )
  ```
  Solves a symmetric/Hermitian positive definite tridiagonal system of linear equations $AX = B$. Error bounds on the solution and a condition estimate are also provided.

-
  ```
  CALL LA_SYSVX / LA_HESVX( A, B, X, UPLO=uplo, AF=af, &
                            IPIV=ipiv, FACT=fact, &
                            FERR=ferr, BERR=berr, &
                            RCOND=rcond, INFO=info )
  ```
  Solves a symmetric/Hermitian/complex indefinite system of linear equations $AX = B$. Error bounds on the solution and a condition estimate are also provided.

- CALL LA_SPSVX / LA_HPSVX(AP, B, X, UPLO=uplo, AFP=afp, &
  IPIV=ipiv, FACT=fact, &
  FERR=ferr, BERR=berr, &
  RCOND=rcond, INFO=info)
  Solves a symmetric/Hermitian/complex indefinite (packed storage) system of linear equations AX = B. Error bounds on the solution and a condition estimate are also provided.

Driver Routines for Linear Least Squares Problems

- CALL LA_GELS(A, B, TRANS=trans, INFO=info)
  Solves over-determined or under-determined linear systems or its transpose, using a QR or LQ factorization of A.
- CALL LA_GELSX(A, B, RANK=rank, JPVT=jpvt, &
  RCOND=rcond, INFO=info)
  Computes the minimum-norm solution to a linear least squares problem, using a complete orthogonal factorization of A.
- CALL LA_GELSS(A, B, RANK=rank, S=s, RCOND=rcond, &
  INFO=info)
  Computes the minimum norm solution to a real linear least squares problem, using the singular value decomposition (SVD) of A.

Driver Routines for generalized Linear Least Squares Problems

- CALL LA_GGLSE(A, B, C, D, X, INFO=info)
  Solves the linear equality-constrained least squares (LSE) problem.
- CALL LA_GGGLM(A, B, D, X, Y, INFO=info)
  Solves a general Gauss-Markov linear model (GLM) problem.

Driver Routines for Standard Eigenvalue and Singular Value Problems

- CALL LA_SYEV / LA_HEEV(A, W, JOBZ=jobz, UPLO=uplo, &

INFO=info)
Computes all eigenvalues and, optionally, eigenvectors of a real symmetric or Hermitian matrix A.

- CALL LA_SPEV / LA_HPEV(AP, W, UPLO=uplo, Z=z, &
  INFO=info)
  Computes all the eigenvalues and, optionally, eigenvectors of a real symmetric / hermitian matrix A in packed storage.

- CALL LA_SBEV / LA_HBEV(AB, W, UPLO=uplo, Z=z, &
  INFO=info)
  Computes all the eigenvalues and, optionally, eigenvectors of a symmetric / Hermitian band matrix A.

- CALL LA_STEV(D, E, Z=z, INFO=info)
  Computes all eigenvalues and, optionally, eigenvectors of a real symmetric tridiagonal matrix A.

- CALL LA_GEES(A, ω, VS=vs, SELECT=select, SDIM=sdim, &
  INFO=info)
  Computes for a non-symmetric matrix A, the eigenvalues, the Schur form T, and, optionally, the matrix of Schur vectors Z. ω is either WR, WI or W.

- CALL LA_GEEV(A, ω, VL=vl, VR=vr, INFO=info)
  Computes for a non-symmetric matrix A, the eigenvalues and, optionally, the left and/or right eigenvectors. ω is either WR, WI or W.

- CALL LA_GESVD(A, S, U=u, VT=vt, WW=ww, JOB=job, &
  INFO=info)
  Computes the singular value decomposition (SVD) of matrix A, optionally computing the left and/or right singular vectors.

Divide and Conquer Driver Routines for Standard Eigenvalue Problems

- CALL LA_SYEVD / LA_HEEVD(A, W, JOBZ=jobz, &
  UPLO=uplo, INFO=info)
  Computes all eigenvalues and, optionally, eigenvectors of a real symmetric or Hermitian matrix A. If eigenvectors are desired, it uses a divide and conquer algorithm.

- CALL LA_SPEVD / LA_HPEVD(AP, W, UPLO=uplo, Z=z, &
  INFO=info)
  Computes all the eigenvalues and, optionally, eigenvectors of a real symmetric / hermitian matrix A in packed storage. If eigenvectors are desired, it uses a divide and conquer algorithm.
- CALL LA_SBEVD / LA_HBEVD(AB, W, UPLO=uplo, Z=z, &
  INFO=info)
  Computes all the eigenvalues and, optionally, eigenvectors of a symmetric / Hermitian band matrix A. If eigenvectors are desired, it uses a divide and conquer algorithm.
- CALL LA_STEVD(D, E, Z=z, INFO=info)
  Computes all eigenvalues and, optionally, eigenvectors of a real symmetric tridiagonal matrix A. If eigenvectors are desired, it uses a divide and conquer algorithm.

Expert Driver Routines for Standard Eigenvalue Problems

- CALL LA_SYEVX / LA_HEEVX(A, W, UPLO=uplo, VL=vl, &
  VU=vu, L=il, IU=iu, M=m, &
  IFAIL=ifail, ABSTOL=abstol, &
  INFO=info)
  Computes all eigenvalues and, optionally, eigenvectors of a real symmetric or Hermitian matrix A. Eigenvalues and eigenvectors can be selected by specifying either a range of values or a range of indices for the desired eigenvalues.
- CALL LA_SPEVX / LA_HPEVX(AP, W, UPLO=uplo, Z=z, VL=vl, &
  VU=vu, IL=il, IU=iu, M=m, &
  IFAIL=ifail, ABSTOL=abstol, &
  INFO=info)
  Computes all the eigenvalues and, optionally, eigenvectors of a real symmetric / hermitian matrix A in packed storage. Eigenvalues/vectors can be selected by specifying either a range of values or a range of indices for the desired eigenvalues.
- CALL LA_SBEVX / LA_HBEVX(AB, W, UPLO=uplo, Z=z, VL=vl, &
  VU=vu, IL=il, IU=iu, M=m, &
  IFAIL=ifail, Q=q, &

ABSTOL=abstol, INFO=info)

Computes all the eigenvalues and, optionally, eigenvectors of a symmetric / Hermitian band matrix A. Eigenvalues and eigenvectors can be selected by specifying either a range of values or a range of indices for the desired eigenvalues.

- CALL LA_STEVX(D, E, W, Z=z, VL=vl, VU=vu, IL=il, IU=iu, &
  M=m, IFAIL=ifail, ABSTOL=abstol, INFO=info)
  Computes all eigenvalues and, optionally, eigenvectors of a real symmetric tridiagonal matrix A. Eigenvalues and eigenvectors can be selected by specifying either a range of values or a range of indices for the desired eigenvalues.

- CALL LA_GEESX(A, ω, VS=vs, SELECT=select, SDIM=sdim, &
  RCONDE=rconde, RCONDV=rcondv, &
  INFO=info)
  Computes for a non-symmetric matrix A, the eigenvalues, the Schur form T, and, optionally, the matrix of Schur vectors Z. Optionally, it also orders the eigenvalues on the diagonal of the real Schur form so that selected eigenvalues are at the top left; computes a reciprocal condition number for the average of the selected eigenvalues, and computes a reciprocal condition number for the right invariant subspace corresponding to the selected eigenvalues. ω is either WR, WI or W.

- CALL LA_GEEVX(A, ω, VL=vl, VR=vr, BALANC=balanc, &
  ILO=ilo, IHI=ihi, SCALE=scale, &
  ABNRM=abnrm, RCONDE=rconde, &
  RCONDV=rcondv, INFO=info)
  Computes for a non-symmetric matrix A, the eigenvalues and, optionally, the left and/or right eigenvectors. Optionally also, it computes a balancing transformation to improve the conditioning of the eigenvalues and eigenvectors (ILO, IHI, SCALE, and ABNRM), reciprocal condition numbers for the eigenvalues (RCONDE), and reciprocal condition numbers for the right eigenvectors (RCONDV). ω is either WR, WI or W.

Driver Routines for Generalized Eigenvalue and Singular Value Problems

- CALL LA_SYGV /LA_HEGV(A, B, W, ITYPE=itype, JOBZ=jobz, &
  UPLO=uplo, INFO=info)
  Computes all the eigenvalues, and optionally, the eigenvectors of a real generalized symmetric-definite or complex Hermitian-definite eigenproblem

- CALL LA_SPGV /LA_HPGV(AP, BP, W, ITYPE=itype, &
  UPLO=uplo, Z=z, INFO=info)
  Computes all the eigenvalues and, optionally, the eigenvectors of a real generalized symmetric-definite eigenproblem.
- CALL LA_SBGV /LA_HBGV(AB, BB, W, UPLO=uplo, Z=z, &
  INFO=info)
  Computes all the eigenvalues, and optionally, the eigenvectors of a real generalized symmetric-definite banded eigenproblem.
- CALL LA_GEGS(A, B, α=alpha, BETA=beta, VSL=vsl, &
  VSR=vsr, INFO=info)
  Computes for a pair of non-symmetric matrices A, B: the generalized eigenvalues (α_r, α_i, β), the Schur form (A, B), and optionally left and/or right Schur vectors (VSL and VSR). α ::= ALPHAR, ALPHAI | ALPHA
- CALL LA_GEGV(A, B, α=alpha, BETA=beta, VL=vl, &
  VR=vr, INFO=info)
  Computes for a pair of non-symmetric matrices A and B, the generalized eigenvalues (α, β), and optionally, the left and/or right generalized eigenvectors. α ::= ALPHAR, ALPHAI | ALPHA
- CALL LA_GGSVD(A, B, ALPHA, BETA, K=k, L=l, U=u, V=v, &
  Q=q, INFO=info)
  Computes the generalized singular value decomposition.

Some Computational Routines for Linear Equations and Eigenproblems

Routines for Linear Equations

- CALL LA_GETRF(A, IPIV, RCOND=rcond, NORM=norm, &
  INFO=info)
  Computes an *LU* factorization of a general rectangle matrix *A* using partial pivoting with row interchanges. Optionally estimates the reciprocal of the condition number if *A* is a square matrix.
- CALL LA_GETRS(A, IPIV, B, TRANS=trans, INFO=info)
  Solves a system of linear equations with a general square matrix *A* using the *LU* factorization computed by LA_GETRF.

- CALL LA_GETRI(A, IPIV, INFO=info)
  Computes the inverse of a matrix using the LU factorization computed by LA_GETRF.

- CALL LA_GERFS(A, AF, IPIV, B, X, TRANS=trans, &
  FERR=ferr, BERR=berr, INFO=info)
  Improves the computed solution X of a system of linear equations $AX = B$ or $A^T X = B$ and provides error bounds and backward error estimates for the solution. LU factors computed by LA_GETRF are used.

- CALL LA_GEEQU(A, R, C, ROWCND=rowcnd, &
  COLCND=colcnd, AMAX=amax, INFO=info)
  Computes row and column scalings intended to equilibrate a rectangle matrix A and reduces its condition number.

- CALL LA_POTRF(A, UPLO=uplo, RCOND=rcond, &
  NORM=norm, INFO=info)
  Computes the Cholesky factorization and optionally estimates the reciprocal of the condition number of a real symmetric or complex Hermitian positive definite matrix A.

Routines for Eigenproblems

- CALL LA_SYGST / LA_HEGST(A, B, ITYPE=itype, &
  UPLO=uplo, INFO=info)
  Reduces a real symmetric-definite or complex Hermitian-definite generalized eigenproblem to standard form.

- CALL LA_SYTRD / LA_HETRD(A, TAU, UPLO=uplo, INFO=info)
  Reduces a real symmetric or complex Hermitian matrix A to real symmetric tridiagonal form T by an orthogonal or unitary similarity transformation: $Q^H A\, Q = T$.

- CALL LA_ORGTR / LA_UNGTR(A, TAU, UPLO=uplo, INFO=info)
  Generates a real orthogonal / complex unitary matrix Q which is defined as the product of elementary reflectors, as returned by LA_SYTRD / LA_HETRD.

Matrix Manipulation Routines

- VNORM = LA_ANGE(A, NORM=norm, INFO=info)
  Returns the value of the one norm, or the Frobenius norm, or the infinity norm, or the element of largest absolute value of a complex matrix A.

- CALL LA_LAGGE(A, KL=kl, KU=ku, D=d, ISEED=iseed, &
  INFO=info)
  Generates a general rectangular matrix A, by pre- and post-multiplying a diagonal matrix D with random orthogonal matrices: $A = U\,D\,V$.