Mathematical Modelling and Analysis
Volume 5, 2000, pages 44–54

LAPACK95 – HIGH PERFORMANCE LINEAR ALGEBRA PACKAGE

J. DONGARRA[1] and J. WAŚNIEWSKI[2]

[1] *Department of Computer Science, University of Tennessee and Mathematical Sciences Section, Oak Ridge National Laboratory*

[1] 107 Ayres Hall, Knoxville, TN 37996-1301, USA

[2] *Scientific Computing Group, Danish Computing Centre for Research and Education (UNI•C)*

[2] DTU, Building 304, DK-2800 Lyngby, Denmark

Received September 28, 1999

ABSTRACT

LAPACK95 is a set of FORTRAN95 subroutines which interfaces FORTRAN95 with LAPACK.
All LAPACK driver subroutines (including expert drivers) and some LAPACK computationals have both generic LAPACK95 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 95 to denote the new FORTRAN 95 interface which is describe here.

We give the background information and references of LAPACK, ScaLAPACK, FORTRAN 95 and HPF in this section. The end of this section contains very brief statements of LAPACK95 too.

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.

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 as possible of the computation is performed by calls to the Basic Linear Algebra Subprograms[14] (BLAS). While LINPACK and EISPACK are based on the vector operation kernels of the Level 1 BLAS. LAPACK is designed at the outset 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. Although 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 [14; 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). 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[15] (Basic Linear Algebra Communication Subprograms). LAPACK runs on any machine where the BLAS[14] 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. The name ScaLAPACK is an acronym for Scalable Linear Algebra PACKage, or Scalable LAPACK

For more information on ScaLAPACK and references on BLAS, BLACS, PBLAS, PVM and MPI see [14; 15; 6; 2; 7; 12].

1.3. FORTRAN 95

FORTRAN has always been the principal language used in the fields of scientific, numerical, and engineering. 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. A summary of the new features is:

- 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.

Taken together, the new features contained in FORTRAN 90/95 ensure that the FORTRAN language will continue to be used successfully for a long time to come. 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 [11].

1.4. High Performance FORTRAN (HPF)

FORTRAN is reaching its limitations on the latest generations of high perforce 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 features 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 serious. 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 95, 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 [9].

1.5. LAPACK for FORTRAN 95

All LAPACK driver subroutines (including expert drivers) and some LAPACK computationals have both generic LAPACK95 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 LAPACK95 (LAPACK77) interface requires the user to specify the F95_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 F95_LAPACK is needed in the first case, the LAPACK95 interface package is called. The module F77_LAPACK is needed in the second case, the LAPACK77 package is directly called.

The present implementation of the LAPACK95 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 LAPACK95 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 [10; 13]), and at UNI•C. Several ScaLAPACK subroutines and test programs are interfaced with HPF.

2. INTERFACE BLOCKS FOR LAPACK 77

All LAPACK77 driver subroutines (including expert drivers) and LAPACK77 computationals have generic interfaces. No distinction is made between single

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

Figure 1. Example1: Module F77_LAPACK is used.

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 $A\,X = B$, 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 internally be used as SP. The interface block SP defines the precision, 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 [8]). In this case SGESV replaces LA_GESV.

For more information see references [3; 4; 8].

3. INTERFACE BLOCKS FOR LAPACK 90

```
 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.

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 $A\,X = B$, 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, in this case single precision. The program works in double precision if DP replaces SP.
- **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. In this case

```
 1  PROGRAM EXAMPLE
 2    USE LA_PRECISION, ONLY: WP => SP
 3    USE f77_LAPACK, ONLY: F77GESV => LA_GESV
 4    USE f90_LAPACK, ONLY: F90GESV => LA_GESV
 5    IMPLICIT NONE
 6    INTEGER :: INFO, J, LDA, LDB, N, NRHS
 7    INTEGER, ALLOCATABLE :: IPIV(:)
 8    REAL :: T0, T1, T2
 9    REAL(WP), ALLOCATABLE :: A(:,:), B(:,:)
10    N = 500; NRHS = 2
11    ALLOCATE( A(N,N), B(N,NRHS), IPIV(N) )
12    CALL RANDOM_NUMBER(A)
13    DO J = 1, NRHS; B(:,J) = SUM( A, DIM=2)*J; ENDDO
14    LDA = N; LDB = N
15    CALL CPU_TIME(T0); CALL CPU_TIME(T1); T0 = T1-T0
16    CALL F77GESV( N, NRHS, A, LDA, IPIV, B, LDB, INFO )
17    CALL CPU_TIME(T2)
18    WRITE(*,*) 'INFO and CPUTIME of F77GESV ', INFO, T2-T1-T0
19    CALL CPU_TIME(T1); CALL F90GESV( A, B ); CALL CPU_TIME(T2)
20    WRITE(*,*) 'CPUTIME of F90GESV ', T2-T1-T0
21  END PROGRAM EXAMPLE
```

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

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 $A\,X = B$, where A is a square matrix, and B and X are rectangular matrices.

For more information see references [3; 8; 4].

4. CODE OF LAPACK90 ROUTINES

The code of LAPACK95 interface 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
- end of routine statement

The LA_PRECISION module and the ERINFO subroutine and other LAPACK95 programs are illustrated in [8].

5. LAPACK95 USER CALLABLE ROUTINES

[8] contains a short description of all LAPACK95 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 [8].

ACKNOWLEDGMENTS

This research was partially supported by the Danish Natural Science Research Council through a grant for the EPOS project (Efficient Parallel Algorithms for Optimization and Simulation) 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 95 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. *LAPACK95 - FORTRAN95 version of LAPACK.* Accessible at 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.* University of Tennessee at Knoxville, Technical Report, CS-95-292, May 1995. Accessible at http://www.netlib.org/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] J. Dongarra and J. Waśniewski, High Performance Linear Algebra Package – LAPACK95. In: *Advances in Randomized Parallel Computing, Kluwer Academic Publishers, Combinatorial Optimization Series, P.M. Pardalos and S. Rajasekaran (Eds.)* 1999 and available as the LAPACK Working Note (Lawn) Number 134. Accessible at http://www.netlib.org/lapack/lawns/lawn134.ps

[9] 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.

[10] P.A.R. Lorenzo, A. Müller, Y. Murakamix, 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

[11] M. Metcalf and J. Reid. *FORTRAN 95 Explained.* Oxford, New York, Tokyo, Oxford University Press, 1990.

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

[13] R.C. Whaley. HPF Interface to ScaLAPACK (1997). Accessible at http://www.netlib.org/scalapack/prototype/

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

[15] BLACS (Basic Linear Algebra Communication Subprograms). Accessible at http://www.cs.utk.edu/~rwhaley/Blacs.html

LAPACK95 – DIDELIO NAŠUMO TIESINĖS ALGEBROS ALGORITMŲ PAKETAS

J. DANGARRA, J. WASNIEWSKI

Šiame darbe aprašytas LAPACK 95 paketas, kurį sudaro FORTRAN95 paprogramių rinkinys. Jos skirtos realizuoti FORTRAN 95 interfeisą su standartine LAPACK biblioteka. Pateikiama būtiniausia informacija apie LAPACK, ScaLAPACK, FORTRAN 95 ir HPF bib-

liotekas. Po to trumpai suformuluojami bendri interfeiso sudarymo principai. LAPACK 95 nėra nauja tiesinės algebros algoritmų biblioteka, o tik konverteris, leidžiantis ir FORTRAN 95 programose naudoti LAPACK algoritmus. Parodyta, kaip galima išnaudoti platesnes FORTRAN 95 galimybes lyginant su standartiniu FORTRAN 77. Pateikti interfeisų pavyzdžiai.