# High Performance Linear Algebra Package LAPACK90

Jack Dongarra[1] and Jerzy Waśniewski[2]

[1] 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

[2] 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

**Abstract.** LAPACK90 is a set of LAPACK90 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 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 (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 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 [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 (Parallel Virtual Machine) and/or MPI (Message Passing Interface). ScaLAPACK is a continuation of the LAPACK project (see section 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 (Basic Linear Algebra Communication Subprograms). LAPACK runs on any machine where the BLAS 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 [2].

## 3 FORTRAN 90

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.
- Rarameterization 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 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 [6].

# 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 90, and the **FORALL** statement and 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 [5].

# 5 LAPACK90

## 5.1 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 )
  Module F90_LAPACK is needed in this case.
- CALL LA_GESV( N, NRHS, A, LDA, IPIV, B, LDB, INFO )
  Module F77_LAPACK is needed in this case.

The documentation for LAPACK90 see in [4, 3]. The LAPACK90 library and documentation will successively be updated. The LAPACK 90 Users' Guide is in progress. 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.

### 5.2 ScaLAPACK for HPF

The HPF ScaLAPACK interface project started in several places (see [7, 8]), and at 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.

## 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. 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.
6. M. Metcalf and J. Reid. *FORTRAN 90 Explained.* Oxford, New York, Tokyo, Oxford University Press, 1990.
7. *P.A.R. Lorenzo, A. Mü ller, 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
8. *R.C. Whaley. HPF Interface to ScaLAPACK.* On web: http://www.netlib.org/scalapack/prototype/ (1997).


This article was processed using the LaTeX macro package with LLNCS style