CS - 90 - 118

# Prospectus for an Extension to LAPACK: A Portable Linear Algebra Library for High-Performance Computers

*E. Anderson, C. Bischof, J. Demmel, J. Dongarra, J. DuCroz, S. Hammarling, and W. Kahan*

Computer Science Department
University of Tennessee
Knoxville, TN 37996-1301

and

Mathematical Sciences Section
Oak Ridge National Laboratory
Oak Ridge, TN 37831

September 26, 1991

This work was supported in part by the National Science Foundation, under grant ASC-8715728 and the National Science Foundation Science and Technology Center Cooperative Agreement No. CCR-8809615.

# Prospectus for an Extension to LAPACK: A Portable Linear Algebra Library for High-Performance Computers

E. Anderson C. Bischof J. Demmel J. Dongarra J. DuCroz
S. Hammarling W. Kahan

September 26, 1991

*Abstract:* LAPACK is a linear algebra library for high-performance computers. The first release, targeted at shared memory machines, will be available in mid 1991, and is already attracting significant interest both from the potential user community and from manufacturers. We propose to extend further the usefulness of LAPACK to the user community by:

- adding linear algebra routines to solve new problems
- producing software for high-performance distributed memory machines
- providing C and Fortran 90 versions
- exploiting clean computer arithmetic to compute faster and more reliably, and
- performing systematic performance evaluation.

LAPACK will simplify code development, make codes more portable among machines of different architectures, increase efficiency, and provide tools for evaluating computer performance.

This work was supported in part by the National Science Foundation, under grant ASC-8715728 and the National Science Foundation Science and Technology Center Cooperative Agreement No. CCR-8809615.

# 1 Introduction

Over the past three years we have designed and implemented a linear algebra library called LAPACK for a wide range of shared memory supercomputers. LAPACK is based on the successful LINPACK [2] and EISPACK [13, 7] libraries. We have developed a functional extension of these libraries which is portable and more efficient across the range of large-scale, shared-memory, general-purpose computers. To achieve this, we have redesigned LINPACK and EISPACK to use a set of basic linear algebra subroutines (called the Level 1, 2, and 3 BLAS) which perform basic operations such as scalar-vector, matrix-vector, and matrix-matrix multiplication. These subroutines, especially the matrix-matrix operations, can be optimized for each machine while the Fortran code that calls them remains identical and hence portable across all the machines. This approach lets us extract most of the performance that each machine has to offer, while restricting machine-dependent code to the BLAS and a few integer "tuning parameters."

In addition to developing faster code, we have added new functions, some of which solve new problems (such as computing condition numbers of eigenvalues and eigenvectors), and some of which solve old problems more accurately (the bidiagonal singular value decomposition).

A comprehensive test package for the entire library has also been developed. The test package has drivers which test groups of related subroutines for accuracy and stability. In addition, we have developed a set of performance-measuring routines to test the efficiency of the LAPACK routines and their variants.

We propose to extend the design and implementation of LAPACK in the following ways:

- Add routines for estimating the condition of the generalized eigenproblem, computing the generalized SVD, solving generalized Sylvester and Lyapunov equations, performing low rank updates of most factorizations, and computing eigenvalues and eigenvectors to high accuracy using Jacobi's method
- Develop distributed-memory versions of selected LAPACK routines
- Develop a C language version of the more heavily used routines, construct a Fortran 90 language interface for all the driver programs, and use the standards work of the Parallel Computing Forum for the parts of the package for which the parallelism is not loop-based
- Rewrite selected routines to exploit special properties of computer arithmetic, especially IEEE Standard Floating-Point Arithmetic
- Develop a systematic performance evaluation suite based on LAPACK.

# 2 Detailed Outline of the Project

## 2.1 Expand the scope of the LAPACK package

After the LAPACK project began in 1987, we assembled a detailed outline for the contents of the package (see LAPACK Working Note #5). This design was reviewed by the developers, manufacturers, and research community as work progressed.

We have implemented much of what was outlined in LAPACK Working Note #5, including testing and timing software. Nevertheless, LAPACK Working Note #5 outlined an ambitious project, representing a significant amount of work, and we will not have completed all of it by the end of our current funding. In addition, we discovered new algorithms during the course of the project which were not reflected in Working Note #5.

The major new subroutines which we will include in the LAPACK extension are:

- Condition estimation for the generalized eigenproblem $A - \lambda B$
- Computing the generalized singular value decomposition
- Solving generalized Sylvester and Lyapunov equations
- Computing low rank updates of most factorizations
- Highly accurate eigenroutines and singular value routines based on Jacobi's method.

The first three extensions have important applications in systems and control theory. The latter two extensions are recent discoveries of routines, based on the Jacoby method, which can solve the symmetric positive definite eigenproblem and SVD more accurately than any algorithm in LINPACK or EISPACK. For this reason we will incorporate these routines in the extension even though they are not as fast as other algorithms.

## 2.2 Develop a distributed-memory version of selected routines from LAPACK

The LAPACK effort currently under way is targeted for shared-memory machines. There are two reasons for this restriction. First, current, general-purpose scientific supercomputers use shared-memory. Second, at the time of the initial proposal, the state of research into distributed-memory linear algebra algorithms was not nearly as advanced as for shared memory. In particular, basic issues such as matrix storage format were still being investigated.

Now, however, the time is appropriate to consider a distributed-memory version of the Basic Linear Algebra Subprograms and LAPACK. The fastest general-purpose scientific supercomputers still use shared memory, but extremely fast distributed-memory machines are now being built. These include machines such as the Intel iPSC/860, the BBN TC2000 Butterfly-II and the NCUBE-2. In addition, published research [6, 5, 8, 10] indicates that for sufficiently large matrix problems, a Fortran-style columnwise storage scheme (while not necessarily the best) is not far from the best. Moreover, the performance is not strongly dependent on the communication topology (ring vs. mesh vs. hypercube, for example).

Given this motivation, we propose to design and test distributed-memory versions of the most important LAPACK algorithms: LU decomposition, Cholesky decomposition, symmetric indefinite matrix decomposition, QR decomposition, reduction to tridiagonal form, and reduction to Hessenberg form. Our approach will be to

1. Choose a small set of distributed-memory machines. Our preliminary plan is to use the Intel iPSC/860 and Cogent since they are fast and will be available at Tennessee/Oak Ridge and Berkeley. Cooperating sites will use other machines.

2. Design distributed BLAS for these machines. Much related work has already been done at Oak Ridge [11, 9, 12], and we plan to build on this. Initially we will use a block column-wrapped mapping, at least on those machines with large enough memories to hold several columns in a single memory. That is, block column $j$ (the $j$th group of $b$ adjacent columns) will be stored in processor $j \bmod p$, where $p$ is the number of processors, and $b$ is a machine dependent parameter.

3. Investigate the feasibility of this scheme on SIMD architectures such as Thinking Machine's CM-2, Active Memory Technology's DAP, and MasPar's MP machine, where the data mapping may be different.

4. Perform timing experiments on the existing LAPACK code, comparing results with those from the highly optimized codes for LU decomposition and triangular system-solving already written at Oak Ridge, Caltech, and other places.

We justify these decisions as follows. First, the size of problems people wish to solve will always grow more rapidly than the number of available processors, so it makes sense to consider storage schemes assuming the number of processors is less than the matrix dimension. (In the case of the Connection Machine, we could store one or more matrix columns in a set of processors, and treat those processors as one larger processor.) Second, the current trend is for floating-point speeds to increase more rapidly than communication speeds, so that compute-bound jobs will run more efficiently than communication-bound jobs for the foreseeable future. Third, compute-bound jobs are relatively less sensitive to the processor topology than communication-bound jobs, making our work more general. Thus, it is reasonable to assume that each processor will work on several columns of the matrix. Column-wrapped mapping fits into the column-oriented storage of matrices in Fortran, which was used in LAPACK, and lets us retain the original LAPACK software. It also does load-balancing automatically.

We feel the time is right to develop a *de facto* standard for a distributed-memory version of the BLAS. We see ourselves in a unique position to help in the formation of these tools. Many people have already contributed to this work, and we propose to convene a group of individuals who have been involved in this effort to develop a proposal which would be circulated to the community and evolve into a *de facto* standard for the distributed-memory BLAS.

The end product will be a careful evaluation of the feasibility of using LAPACK with distributed BLAS to get portable, high-performance codes, as well as a number of well-tested codes that would be freely available for others to use. We will promote the design and adoption of a distributed BLAS standard analogous to the successful standards for the Level 2 and 3 BLAS [4, 3].

## 2.3 Implement part of the package in C, design a Fortran 90 language interface, and use standards work of the Parallel Computing Forum

These proposed activities will make LAPACK available to a wider group of users, and encourage the use of standards in exploiting features such as parallel processing.

We plan to produce a version of the library that will be written in the C language. This part of the project will be done in cooperation with a group at AT&T Bell Laboratories in Murray Hill, NJ. That group has developed an automatic translation system that takes in Fortran code and produces a C version of the program. This is not just a simple translator, but one that has some intelligence of how Fortran and C store arrays, and attempts to convert the Fortran array column storage to C array row storage. Our objective is to make the library available to a new class of computational scientists who use C as their development language.

In addition, we plan to take a subset of the library, including the driver routines which solve complete problems using a series of calls to lower level subroutines, and implement it in the Fortran 90 language. This effort will exploit a number of features in the Fortran 90 standard that should simplify the programming and user interface to the package. In particular, we will create an implementation that uses the array features and dynamic storage allocation. We believe that these features will greatly enhance the readability of our software, and reduce the calling sequence. We view this activity as an experiment in the use of the Fortran 90 language for numerical computation, in general, and matrix algorithms, in particular.

Over the past two years a group of vendors and university researchers have defined a standard set of extensions to Fortran to support parallel processing. This group, called the Parallel Computing Forum, has produced a document that describes a set of constructs for parallel processing. This effort has been transferred to ANSI X3H5 to propose a standard. We intend to implement a subset of the LAPACK routines in this proposed standard. The routines to be implemented are ones that cannot easily be expressed with the simple notion of loop based parallelism, such as multisectioning and the divide-and-conquer algorithm for the symmetric eigenvalue problem and the singular value decomposition.

## 2.4 Exploit special properties of computer arithmetic

The current LAPACK effort, like many other library projects, attempts to be as portable as possible across a wide range of computer architectures. One price paid for this portability is the need to function correctly on machines with very different kinds of floating point arithmetic—including Cray arithmetic, noted for its anomalous behavior, and IEEE arithmetic, with its carefully constructed semantics. As a result, the LAPACK algorithms assume properties of the arithmetic available on all of the machines; in short, they are programmed assuming the worst possible arithmetic.

We have identified a number of algorithms within LAPACK that would benefit greatly from sacrificing some portability in order to exploit particular properties of computer arithmetic. These benefits include enhanced speed, robustness, and accuracy. We propose to rewrite these algorithms to reap these advantages. Thus, just as the BLAS are machine-optimized kernels designed to exploit each architecture invisibly to the user, our codes would exploit properties of

the arithmetic invisibly to the user.

There are various floating point arithmetics available, satisfying a variety of different assumptions about their accuracy. We plan to explore several of these different arithmetics, with the goal of measuring exactly how much benefit one obtains from each additional assumption. Thus the new algorithms, although not completely portable, will still work on a wide variety of important machines (which we can easily identify), and so enhance speed, robustness, and accuracy for those machines. Manufacturers who have expended the effort to design careful arithmetic will be repaid by having good library software that exploits it. Currently, such good library software is rare, making these carefully designed machines relatively less attractive to potential customers.

In particular, IEEE floating-point arithmetic provides the best environment for writing such specialized codes. IEEE arithmetic has been widely adopted in the microprocessor industry (by Intel, Motorola, Weitek, MIPS, Sun, and others), and is beginning to appear in supercomputers (the Tera Machine and Steve Chen's Supercomputer Systems, Inc., machine will conform to all or a large part of the IEEE standard, and Cray has stated it will eventually adopt IEEE arithmetic). Not all of the algorithms we propose require IEEE arithmetic, but all will run using IEEE arithmetic.

It would clearly be advantageous to be able to take advantage of special arithmetic properties, without sacrificing portability. To accomplish this, it is necessary to be able to discover the relevant arithmetic properties efficiently and portably at run-time. To support this, we will develop a standard set of environmental inquiry subroutines which will supply the necessary information. Like the BLAS, they can be implemented by each manufacturer in a machine-specific way. We expect these subroutines to be very short and easy to write for each specific machine, but they will make it much easier to make highly robust numerical codes widely available.

In addition we expect our codes to become models to teach others to write highly robust, expert numerical codes.

### 2.5 Conduct systematic performance evaluation

As participants in the LAPACK effort, we have had access to a wide variety of high-performance computers. In developing the package we have gathered timing statistics for the programs and their variants across a wide range of machines. This information has guided us in the improvement and selection of the algorithms and software that will be included in the collection. We are now in a position to use the information and techniques developed in the first phase of the LAPACK project to evaluate and predict the performance of matrix algorithms.

The need for systematic performance evaluation techniques has been emphasized by the National Research Council in their report, "An Agenda for Improved Evaluation of Supercomputer Performance" [1]. The Council recommends that researchers develop procedures for characterizing the applications and algorithms in terms of "fundamental units of computations," and devise techniques for composing operations of data structures into parameterized algorithm elements.

In response to this recommendation, we propose to develop a scheme for the analysis of matrix

algorithms common to linear algebra. Our approach will involve measuring the performance of a set of operations basic to linear algebra, and determining how the algorithm should be structured relative to those operations. Specifically, we will (1) isolate the fundamental operations (such as matrix-multiply, data interchanges, and subroutine overhead) for the LAPACK algorithms; (2) identify the parameters needed for each of these elements; and (3) develop a performance model of the complete algorithm which can be calibrated and used to collect actual performance data.

We will start with the shared-memory model. As our base of algorithms expands on distributed-memory machines, so will the performance evaluation techniques.

The goal of this effort is to develop techniques and software required to understand and predict the performance of various algorithms on different hardware by measuring a small set of basic operations. As an outcome of this project, we will identify the strengths and limitations of various architectures for scientific computing, develop techniques and tools for measuring and tuning numerical software, and construct methods for performance prediction.

# 3 Scope of the Proposed Project and Future Work

We have been encouraged by the way vendors and the computational community as a whole have embraced our efforts. We anticipate a cycle of effort, with experience gained on some machines being incorporated into a library and disseminated widely, leading to new experiences and improvements.

# 4 Potential Contribution to Education and Knowledge Transfer

The LAPACK project has had an excellent record of knowledge transfer among the developers, members of the numerical community, and manufacturers of high-performance computers. Publication is the primary method of disseminating knowledge gained from scientific and technological research. Since the project started in 1987, we have produced twenty-five working notes which explain and teach many of the important techniques developed in the project. Another indicator of knowledge transfer is the number of commercial hardware and software firms that have participated in our project through the testing, verification, and timing of our software. We intend to continue this emphasis on knowledge transfer in the development of the enhanced LAPACK library. Several commercial vendors have already indicated their willingness to incorporate our software into their libraries.

Education has also been an important goal of the LAPACK project. The project has encouraged participation by students in the testing and evaluation of the package, and has presented short courses and workshops to potential users. We have had a high level of success in getting the manufacturers to implement efficient versions of the Level 1, 2, and 3 BLAS. The research project proposed here will continue this emphasis on education. We anticipate the participation of a large number of students, particularly in the implementation and testing of the new software systems. Since the research involves components of science, computational mathematics, and computer science, participation will lead to an extraordinarily broad background for the

students involved. Students will also be exposed to a broad variety of computing resources to which they otherwise might not have access.

As its highly successful predecessor LINPACK has done, we expect LAPACK to play a major role in developing courses designed around numerical computations and mathematical software. We plan to structure an LAPACK Users' Guide in such a fashion that it can be used both for classroom instruction and for reference.

## 5 Organization

The project is a joint effort of a number of researchers at several institutions: University of Tennessee, University of California, Berkeley, Numerical Algorithms Group, Ltd., and others. The first two groups have received funding from the National Science Foundation; the Numerical Algorithms Group in the United Kingdom has agreed to cooperate in this research effort. The three institutions plan to work in a coordinated but independent fashion. The University of Tennessee will serve as the center for the project and will be responsible for collecting, editing, testing the material, and distributing the software. We plan to meet at least twice a year to coordinate activities and to discuss developments.

We shall use testsites to help in testing out ideas and in verifying that the software is working correctly. The testsites will, by necessity, receive pre-release versions of the software. Once the programs and test material are ready for distribution to testsites, we will release preliminary copies of the package to those people having a legitimate research interest.

We also plan to continue distributing working notes to anyone interested and to report at conferences on the status, progress, and future directions throughout the project life. The notes are intended to be a means by which tentative decisions on technical matters can be disseminated for comment. A secondary purpose of the notes is to serve as an archive detailing our progress. It should be stressed that much of the material in the notes will be of a tentative nature and will not represent a commitment to any specific action.

We encourage anyone with questions or comments about the project to contact any of the principal investigators on the project. Our plan is to keep the project open to the user community. We believe we can benefit from the advice and exchange among our colleagues.

## References


[1] An agenda for improved evaluation of supercomptuer performance. National Research Council, National Academy Press, 1986.

[2] J. Dongarra, J. Bunch, C. Moler, and G. W. Stewart. *LINPACK User's Guide*. SIAM, Philadelphia, PA, 1979.

[3] J. Dongarra, J. Du Croz, I. Duff, and S. Hammarling. A set of level 3 basic linear algebra subprograms. *ACM Transactions on Mathematical Software*, 16:1–17, 1990.

[4] J. Dongarra, J. Du Croz, S. Hammarling, and R. Hanson. An extended set of fortran basic linear algebra subroutines. *ACM Transactions on Mathematical Software*, 14(1):1–17, March 1988.

[5] G. Fox. Square matrix decomposition — Symmetric, local, scattered. CalTech Publication Hm-97, California Institute of Technology, Pasadena, CA, 1985.

[6] G. Fox and S. Otto. Algorithms for concurrent processors. *Physics Today*, 37(5):50–59, 1984.

[7] B. S. Garbow, J. M. Boyle, J. J. Dongarra, and C. B. Moler. *Matrix Eigensystem Routines – EISPACK Guide Extension*, volume 51 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin, 1977.

[8] A. Geist and M. Heath. Parallel Cholesky factorization on a hypercube multiprocessor. Technical Report ORNL-6190, Oak Ridge National Laboratory, August 1985.

[9] A. Geist and M. Heath. Matrix factorization on a hypercube multiprocessor. In M. Heath, editor, *Hypercube Multiprocessors, 1986*, pages 161–180, Philadelphia, PA, 1986. Society for Industrial and Applied Mathematics.

[10] A. George, M. Heath, and J. Liu. Parallel Cholesky factorization on a shared memory multiprocessor. *Lin. Alg. & Appl.*, 77:165–187, 1986.

[11] M. Heath. Hypercube applications at Oak Ridge National Laboratory. In M. Heath, editor, *Hypercube Multiprocessors, 1987*, pages 395–417, Philadelphia, 1987. Society for Industrial and Applied Mathematics.

[12] M. Heath and C. Romine. Parallel solution of triangular systems on distributed-memory multiprocessors. *SIAM J. Sci. Statist. Comput.*, 9(3):558–588, May 1988.

[13] B. T. Smith, J. M. Boyle, J. J. Dongarra, B. S. Garbow, Y. Ikebe, V. C. Klema, and C. B. Moler. *Matrix Eigensystem Routines – EISPACK Guide*, volume 6 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin, 1976.