ARGONNE NATIONAL LABORATORY
9700 South Cass Avenue
Argonne, Illinois 60439

Prospectus for the Development of a Linear Algebra Library for High-Performance Computers

James Demmel, Jack J. Dongarra, Jeremy Du Croz, Anne Greenbaum, Sven Hammarling, and Danny Sorensen

Mathematics and Computer Science Division

Technical Memorandum No. 97

September 1987

.

ABSTRACT

We propose to design and implement a transportable linear algebra library in Fortran 77 for efficient use on high-performance computers. The library is intended to provide a uniform set of subroutines to solve the most common linear algebra problems and to run efficiently on a wide range of architectures. This library, which will be freely accessible via computer network, not only will ease code development, make codes more portable among machines of different architectures, and increase efficiency, but also will provide tools for evaluating computer performance. The library will be based on the well-known and widely used LINPACK and EISPACK packages for linear equation solving, eigenvalue problems, and linear least squares. LINPACK and EISPACK have provided an important infrastructure for scientific computing on serial machines, but they were not designed to exploit the profusion of parallel and vector architectures now becoming available. We propose to restructure the algorithms in terms of calls to a small number of extended Basic Linear Algebra Subroutines each of which implements a basic operation such as matrix multiplication, rank-k matrix updates, and the solution of triangular systems. These operations can be optimized for each architecture, but the underlying numerical algorithms will be the same for all machines.

Prospectus for the Development of a Linear Algebra Library for High-Performance Computers

James Demmel †

Computer Science Department
Courant Institute
251 Mercer Street
New York, New York 10012

Jack J. Dongarra †

Mathematics and Computer Science Division
Argonne National Laboratory
Argonne, Illinois 60439-4844

Jeremy Du Croz

Numerical Algorithms Group Ltd.
NAG Central Office, Mayfield House
256 Banbury Road, Oxford OX2 7DE, England

Anne Greenbaum †

Computer Science Department
Courant Institute
251 Mercer Street
New York, New York 10012

Sven Hammarling

Numerical Algorithms Group Ltd.
NAG Central Office, Mayfield House
256 Banbury Road, Oxford OX2 7DE, England

Danny Sorensen ‡

Mathematics and Computer Science Division
Argonne National Laboratory
Argonne, Illinois 604394844

† Work supported in part by the National Science Foundation.

‡ Work supported in part by the Applied Mathematical Sciences subprogram of the Office of Energy Research, U. S. Department of Energy, under contract W-31-109-Eng-38.

Typeset on November 17, 1987.

1. Introduction and Objectives

We propose to design and implement a transportable linear algebra library in Fortran 77 for efficient use on a wide range of high-performance computers. The production of such a library for the most commonly encountered problems of linear algebra would have several benefits:

(1) It would facilitate the development of scientific codes on high-performance computers. This area was recently identified by the Computational Science and Engineering Initiative of the National Science Foundation as in serious need of development. The large and growing variety of machine architectures puts a heavy burden on the scientific programmer to use each machine efficiently, since speed is the major reason to use high performance computers. The availability of a highly efficient library for standard linear algebra problems on each major machine would free the programmer to work on more interesting parts of the code.

(2) It would increase the portability of scientific codes between different computing environments. Programs written largely in terms of calls to a standard library would require less work to tune to the new computer architecture, since the library routines would already be tuned.

(3) It would improve the utilization of a scarce resource. By making efficient, state-of-the-art codes available even to beginning users, more efficient use could be made of expensive supercomputer cycles.

(4) It would provide tools to aid performance evaluation of computers. A national study [1] has identified the evaluation of supercomputer performance as an area in need of development and standardization.

To realize these benefits, the new library must satisfy several criteria. First, the library must be highly efficient, or at least "tunable" to high efficiency, on each machine. Otherwise it will not be useful for benchmarking, nor will it improve utilization; and users will continue to write their own (not necessarily better) algorithms. Second, the user interface must be uniform across machines. Otherwise much of the convenience of portability would be lost. Third, the programs must be widely available. The success of the NETLIB facility [6] has demonstrated how useful and important it is for these codes to be available easily, and preferably on line. We propose to distribute the new library in a similar way, for no cost or a nominal cost only. In addition, the programs must be well documented, in the style of the LINPACK manual [2].

To achieve these goals, we propose a linear algebra library, based on the successful EISPACK [11, 9] and LINPACK [2] libraries, with the following further developments:

- integration of the two sets of algorithms into a unified library, with a systematic design;

- incorporation of recent algorithmic improvements; and

- restructuring of the algorithms to make as much use as possible of the Basic Linear Algebra Subprograms (BLAS). Use of the BLAS is the basis of our approach to achieving efficiency, and is discussed at greater length in Section 2.2.

In short, such a library would become a central part of the infrastructure of a growing high-performance scientific programming environment, much as conventional libraries for serial machines are essential to conventional scientific computing.

Section 2 describes technical aspects of the project, and Section 3 the proposed organization.

2. Outline of the Project

2.1 Contents

The new library will provide approximately the same functionality as LINPACK and EISPACK together, namely, solution of systems of simultaneous linear equation, least-squares solution of over-determined systems of equations, and solution of matrix eigenvalue problems (standard and generalized). The associated matrix factorizations (LU, Cholesky, QR, SVD, Schur, generalized Schur) will also be provided, as will related computations such as reordering of the factorizations and condition numbers (or estimates thereof). Dense and band matrices will be provided for, but not general sparse matrices. In all areas, similar functionality will be provided for real and complex matrices.

Many of the algorithms of LINPACK and EISPACK will be carried over with only minor modifications to their numerical behavior, although often with extensive restructuring of the code in order to improve performance (see Section 2.2). Some algorithms may be deleted, especially where there is duplication or overlap in functionality between the contents of the two packages or if they are no longer thought to be useful. Other algorithms may be extended to provide additional functionality. Where the state of the art is sufficiently clear, new algorithms will be added. In some cases the relative merits of competing algorithms will be re-examined in the light of their performance on modern high-performance computers.

Test software will be developed, building on the test software already developed for EISPACK, LINPACK, and the BLAS. Timing programs will be provided to measure efficiency.

2.2 Use of the BLAS

Our approach to achieving high efficiency will be based on the use of a limited set of the Basic Linear Algebra Subprograms (BLAS).

LINPACK was constructed using the original set of BLAS, which perform low-level operations such as dot-products and adding of the multiple of one vector to another [10]. We shall refer to these BLAS as Level 1 BLAS. Efficient implementations of the Level 1 BLAS enabled LINPACK routines to achieve high efficiency on some of the most powerful scalar machines. However, the granularity of the operations is too small for effective use of the power of most vector or parallel machines.

More recently, higher level BLAS have been specified which perform operations of larger granularity, namely, matrix-vector operations (Level 2 BLAS [4]) and matrix-matrix operations (Level 3 BLAS [3]): for example, matrix-vector (or matrix-matrix) multiply-and-add, rank-one (or rank-k)

matrix update, or solution of a triangular system (or systems) of equations. Optimized implementations of Level 2 BLAS can come close to achieving the maximum possible efficiency on a large class of uniprocessor vector machines. However, their performance is still well below the best possible on multiprocessor machines and on machines where the performance is limited by the data traffic between different levels of memory. Better performance on these machines can be achieved by the use of Level 3 BLAS.

We intend to code the routines in the new library so that as much of the computation as possible is performed by calls to Level 2 or Level 3 BLAS. Efficiency can then be achieved by linking the routines to optimized implementations of the BLAS.

We do not regard the provision of optimized implementations of the BLAS as part of this project, although we shall do as much as we can to encourage and assist such provision. Specifications, model implementations, and test programs are already available. We hope that vendors or other users will take up the challenge of optimizing the performance of the Level 2 or Level 3 BLAS on specific machines. Ideally, vendors of high-performance computers will supply optimized BLAS as part of their run-time libraries delivered with every machine (similar to SCILIB on CRAY machines). (For the Level 2 BLAS this is already beginning to happen. The Level 3 BLAS have been specified very recently, and their specification is still to some extent provisional.)

The scope for using Level 2 or Level 3 BLAS varies among the different algorithms that are proposed. In some algorithms (e.g. computing eigenvalues of a symmetric tridiagonal matrix) there is no scope at all. In the majority of the algorithms there is certainly scope for using Level 2 BLAS, and a considerable amount of experience of doing so has already been accumulated [5, 7, 8].

To exploit Level 3 BLAS, one usually must restructure the algorithm into a "block" form, in which the original matrices are partitioned into submatrices or blocks, and the algorithm is expressed in terms of basic matrix-matrix operations on the blocks. Many authors have demonstrated the effectiveness of block algorithms on many of our target machines (see the references cited in [3]). The performance of the algorithms depends on the dimensions chosen for the blocks. It will therefore be necessary to investigate the appropriate blocking strategy for each of our target machines, and then develop a mechanism whereby the routines can determine good block dimensions automatically (possibly via a machine-specific enquiry function).

Block algorithms generally require an unblocked version of the same algorithm to be available to operate on a single block. Therefore, the development of the software will fall naturally into two phases: first, develop unblocked versions of the routines, calling the Level 2 BLAS wherever possible; then develop blocked versions where possible, calling the Level 3 BLAS.

It is likely that this project will reveal the need for a few additional basic routines whose performance may need to be optimized for different architectures and may be regarded as extensions to the current sets of BLAS (e.g., applying sequences of plane rotations to a matrix).

We believe that if we are developing the library in Fortran 77 (see Section 2.4), the only reasonable means of identifying matrix-vector or matrix-matrix operations within the algorithms is to write explicit calls to BLAS-type routines. It is not likely that many compilers will be able to recognize operations of this level of granularity if coded in standard Fortran 77. Array syntax, as proposed for Fortran 8X, would be preferable (at least for some operations) but is not available except as non-standard features on a few machines.

Both the algorithms in the library and the higher level BLAS from which they would be constructed would serve as benchmarks for supercomputer performance evaluation. A report from the Committee on Supercomputer Performance and Development[1] has recommended using a range of routines from program kernels (like the BLAS) and basic routines (like LINPACK and EISPACK) to large application codes (such as radiation transport and fluid dynamics) for benchmarks. Our proposed effort would supply useful routines for this work. The new library would also provide a lower bound on performance, which any competing algorithm would have to exceed.

2.3 Target Machines

The class of machines on which the library will be designed to perform efficiently consists of machines with one or more processors, each of which has a powerful vector-processing capability. It is assumed that the number of such processors is modest (no more than 20, say). This class of machines includes all of the most powerful machines that are currently available and in use for general-purpose scientific computing: CRAY-2, CRAY X-MP, CYBER 205, ETA-10, Fujitsu/Amdahl VP, IBM 3090/VF, NEC SX, Alliant FX/8, Convex C-1, and Scientific Computer Systems SCS-40.

We do not claim that the strategy of using Level 2 or Level 3 BLAS will necessarily attain optimal performance on all these machines; indeed, some algorithms can be structured in several different ways, all calling Level 3 BLAS, but with different performance characteristics. In such cases we shall aim to choose the structure that provides the best ‘‘average’’ performance over the range of target machines.

We hope that the library will also perform well on a wider class of parallel machines, including the IBM RP3, BBN Butterfly, Sequent Symmetry, Encore Multimax, and FPS T-Series.

On conventional serial machines, the performance of the library is expected to be at least as good as that of the current LINPACK and EISPACK codes. Thus the library will be available and suitable for use across the whole range of machines from personal computers to supercomputers to experimental architectures.

2.4 Programming Language and Style

The software will be developed in standard Fortran 77, using extensions to the standard only where necessary.

Single- and double-precision versions will be prepared; conversion between different precisions will be performed automatically by software tools.

Routines for complex matrices will use the COMPLEX data type (like LINPACK, but unlike EISPACK); hence the availability of a double-precision complex (COMPLEX*16 or DOUBLE

COMPLEX) data type will be assumed as an extension to Fortran 77. Routines for real and complex matrices will be written in such a way as to maintain a close correspondence between the two, and to permit automatic transformation, as far as possible; however, in some algorithms (e.g., unsymmetric eigenvalue problems) the correspondence will necessarily be weaker.

The code will be written to permit as much automatic vectorization as possible. Machine-specific compiler directives may be required to communicate vectorizability or parallelism. We shall also take care to make the code safe for use in multitasking applications (i.e., avoiding the use of COMMON or SAVEd variables). Our aim is to make the code entirely machine-independent if possible, and otherwise to use software tools to generate variant versions, or to limit machine-dependencies to a few simple enquiry routines.

It is regrettable that Fortran 8X is not yet available to implement the library. Fortran 8X is likely to have a number of features that would improve the design and coding of the library—optional arguments, dynamic allocation of workspace, and array features, to name a few. However, if we are to begin development and testing of the library now on our range of target machines, there seems no reasonable choice other than Fortran 77.

2.5 User Interface and Documentation

The user interface to the routines will be similar in style to that of LINPACK. Routine names will follow a systematic scheme, and arguments will appear in a systematic order.

We intend to provide a set of top-level driver routines that can solve a complete problem (e.g., compute all eigenvalues and eigenvectors of a symmetric matrix). The top-level routines will each call a number of lower level routines which perform the parts of the solution (e.g., reduction to tridiagonal form, accumulation of the orthogonal transformation matrix, compute eigenvectors and eigenvectors of a tridiagonal matrix).

Both levels of routines will be accessible to users and will be documented in a guide, similar in many respects to the LINPACK Users' Guide. The guide will be divided into two parts: the first will describe the top-level problem-solving routines, and the second will describe the lower-level component routines and the algorithmic details.

3. Organization

This project is an outgrowth and a natural transition from the EISPACK and LINPACK packages that have been developed over the last 15 years. The project is a joint effort of a number of researchers at several institutions: Argonne National Laboratory, Courant Institute, and Numerical Algorithms Group Ltd. 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. Argonne will serve as a center for the project and be responsible for collecting, editing, testing material, and distributing the software. We intend to meet at least twice a year to coordinate activities and discuss developments.

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

We also plan to distribute 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 the 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 outside. We believe we can benefit from the advice and exchange among our colleagues.

References

1. ‘‘An Agenda for Improved Evaluation of Supercomputer Performance,’’ National Research Council, 1986.

2. J.J. Dongarra, J. Bunch, C. Moler, and G. Stewart, *LINPACK Users’ Guide*, SIAM Pub., Philadelphia, 1979.

3. J.J. Dongarra, J. DuCroz, I. Duff, and S. Hammarling, ‘‘A Proposal for a Set of Level 3 Basic Linear Algebra Subprograms,’’ Argonne National Laboratory Report, ANL-MCS-TM-88, April 1987.

4. J.J. Dongarra, J. DuCroz, S. Hammarling, and R. Hanson, ‘‘An Extended Set of Fortran Basic Linear Algebra Subprograms,’’ Argonne National Laboratory Report, ANL-MCS-TM-41 (Revision 3), November 1986.

5. J.J. Dongarra and S. C. Eisenstat, ‘‘Squeezing the Most out of an Algorithm in Cray Fortran,’’ *ACM Trans. Math. Software*, vol. 10, 3, pp. 221-230, 1984.

6. J.J. Dongarra and E. Grosse, ‘‘Distribution of Mathematical Software via Electronic Mail,’’ *Comm of the ACM*, vol. 30,5, pp. 403-407, July, 1987 .

7. J.J. Dongarra, L. Kaufman, and S. Hammarling, ‘‘Squeezing the Most Out of High Performance Computers for Finding the Eigenvalues,’’ *Linear Algebra and Its Applications*, vol. 77, pp. 113-136, 1986.

8. J.J. Dongarra and D.C. Sorensen, ‘‘Linear Algebra on High-Performance Computers,’’ in *Proceedings Parallel Computing 85*, ed. U. Schendel, pp. 3-32, North Holland, 1986.

9. B.S. Garbow, J.M. Boyle, J.J. Dongarra, and C.B. Moler, *Matrix Eigensystem Routines - EISPACK Guide Extension*, 51, pp. Springer-Verlag, Lecture Notes in Computer Science, 1977.

10. C. Lawson, R. Hanson, D. Kincaid, and F. Krogh, ‘‘Basic Linear Algebra Subprograms for Fortran Usage,’’ *ACM Transactions on Mathematical Software*, vol. 5, pp. 308-323, 1979.

11. B.T. Smith, J.M. Boyle, J.J. Dongarra, B.S. Garbow, Y. Ikebe, V. Klema, and C. Moler, *Matrix Eigensystem Routines - EISPACK Guide, Second Edition*,, 6, pp. Springer-Verlag, Lecture Notes in Computer Science, 1976.