Case Studies on The Development of ScaLAPACK and the NAG Numerical PVM Library

J. J. Dongarra, The University of Tennessee at Knoxville and Oak Ridge National Laboratory
S. Hammarling, The University of Tennessee at Knoxville and The Numerical Algorithms Group
A. Petitet, The University of Tennessee at Knoxville
The University of Tennessee at Knoxville, Computer Science Department, 104 Ayres Hall, Knoxville, TN 37996-1301, USA
Oak Ridge National Laboratory, Mathematical Sciences Section, P.O.Box 2008, Bldg. 6012, Oak Ridge, TN 37831-6367, USA
The Numerical Algorithms Group Ltd, Wilkinson House, Jordan Hill Road, Oxford, OX2 8DR, UK

Abstract

In this paper we look at the development of ScaLAPACK, a software library for dense and banded numerical linear algebra, and the NAG Numerical PVM Library, which includes software for dense and sparse linear algebra, quadrature, optimization and random number generation. Both libraries are aimed at distributed memory machines, including networks of workstations.

The paper concentrates on the underlying design and the testing of the libraries.

Keywords

BLAS, LAPACK, NAG, numerical libraries, numerical linear algebra, numerical software, ScaLAPACK

1 INTRODUCTION

This paper looks at the development of two software libraries aimed at distributed memory machines, ScaLAPACK* and the NAG Numerical PVM Library†, concentrating on the underlying design of the libraries and issues concerned with testing the components of the libraries.

The development of ScaLAPACK, and related software, had a strong influence on the NAG Numerical PVM Library and so much of the paper will concentrate on the development of ScaLAPACK.

ScaLAPACK itself is built upon LAPACK‡, and so the paper will start with a brief discussion of the LAPACK project.

The LAPACK project has produced a software package, also called LAPACK which stands for Linear Algebra PACKage, for dense and banded linear algebra problems targeted at high performance shared memory machines. Part of the ScaLAPACK project is concerned with porting LAPACK to distributed memory machines, and has produced the software package ScaLAPACK, which stands for Scalable Linear Algebra PACKage. There has been a new release of each of these packages in 1996, and a further release of ScaLAPACK is planned for the autumn of 1996.

LAPACK, ScaLAPACK and the NAG Numerical PVM Library are all intended to be portable and efficient in their target environments, as well as being maintainable, and we shall discuss the infrastructure that was required to achieve these attributes.

LAPACK has built upon the development of the BLAS§ (Basic Linear Algebra Subprograms), which are used to achieve efficiency by performing the computationally intensive operations, and so form the portability layer for LAPACK. ScaLAPACK, which uses an SPMD message passing programming paradigm, utilizes the PBLAS¶ (Parallel BLAS) for its computationally intensive parts and the BLACS‖ (Basic Linear Algebra Communication Subprograms) for communication; the BLACS being the communication equivalents of the BLAS. The PBLAS themselves call the single node BLAS for computation and the BLACS for communication; indeed most of the calls to the BLACS from ScaLAPACK are made within the PBLAS. Thus the PBLAS are effectively the portability layer for ScaLAPACK.

This structure has enabled the ScaLAPACK software to closely resemble the LAPACK software, thus considerably aiding the porting process. Additionally, the interfaces for the ScaLAPACK routines have been made as close as possible to the LAPACK routines, with the intention of making the porting of users programs straightforward.

The NAG Numerical PVM Library incorporates much of ScaLAPACK, uses the same

*`http://www.netlib.org/scalapack/index.html`

†`http://www.nag.co.uk`

‡`http://www.netlib.org/lapack/index.html`

§`http://www.netlib.org/blas/index.html`

¶`http://www.netlib.org/scalapack/html/pblas_qref.html`

‖`http://www.netlib.org/blacs/index.html` programming model, and similarly utilizes the PBLAS and BLACS as much as possible elsewhere.

The paper will include discussion of the development of the PBLAS and the BLACS, will look at the additional requirements for testing the libraries in a distributed memory environment, and will mention the challenges of making the ScaLAPACK and NAG Numerical PVM Library reliable in a heterogeneous computing environment.

2 THE BLAS

Fundamental to the attempt to develop both portable and efficient software has been the specification of the Level 1, Level 2 and Level 3 BLAS (Basic Linear Algebra Subprograms; Lawson, Hanson, Kincaid & Krogh (1979), Dongarra, Du Croz, Hammarling & Hanson (1988) and Dongarra, Du Croz, Duff & Hammarling (1990)), and the implementation of efficient versions by manufacturers and others on high performance machines. The BLAS encapsulate a set of computational kernels for numerical linear algebra and many of the algorithms of numerical linear algebra have been adapted to utilize these BLAS. LINPACK (Dongarra, Bunch, Moler & Stewart 1978), a widely used high quality public domain software package for the solution of dense and banded linear systems developed in the late 1970s, made extensive use of the Level 1 BLAS, the Level 2 BLAS have been successfully exploited on vector supercomputers (see for example Hammarling (1993) and the references given there), and the Level 3 BLAS have been widely used in software, such as LAPACK, aimed at shared memory machines with a hierarchy of memory and possibly multiple processors.

It is important to think of the BLAS as providing specifications, the intention being for manufacturers, or others, to provide tuned implementations on particular machines. Vanilla Fortran 77 versions are available from netlib** (Dongarra & Grosse 1987, Dongarra, Rowan & Wade 1993), but these are intended to be model implementations, rather than tuned versions for any specific machine.

To support and encourage the use of the BLAS and their implementation a set of test programs are also supplied on netlib, designed to ensure that implementations conform to the specification and have been correctly installed. The test programs for the Level 2 and Level 3 BLAS have the following features:

- The parameters of the test problems and the names of the subprograms to be tested are specified by means of a data file, which can easily be modified for debugging.
- The data for the test problems are generated internally and the results are checked internally.
- The programs check that no input arguments are changed by the routines except the designated output vector or matrix.
- All error exits (caused by illegal argument values) are tested.
- The programs generate a *history* or *snapshot* file as an additional debugging aid.

**`http//:www.netlib.org`

An aspect that with hindsight should, perhaps, have been emphasized and more precisely described in the specifications, is that the BLAS are also expected to be numerically accurate and so naturally the test programs examine the accuracy of the BLAS. A test ratio is determined by scaling the error bounds by the inverse of machine precision, ϵ^{-1}. This ratio is compared with a constant threshold value defined in the input data file. Test ratios greater than the threshold are flagged as *suspect*. On the basis of experience a threshold value of 16 is recommended, but the precise value is not critical. Errors in the routines are most likely to be errors such as errors in array indexing, which will almost certainly lead to a totally wrong result. A more subtle potential error is the use of a single precision variable in a double precision computation. This is likely to lead to a loss of half the machine precision. The test programs regard a test ratio greater than $\epsilon^{-\frac{1}{2}}$ as an *error*.

It seems fair to say that the BLAS have been very successful in achieving their aim and are widely implemented on todays workstations, high-performance shared memory machines and on single nodes of distributed memory parallel machines.

3 LAPACK

The work on the BLAS and on adapting algorithms to utilize the BLAS has culminated in the development of the dense linear algebra package, LAPACK (Anderson, Bai, Bischof, Demmel, Dongarra, Du Croz, Greenbaum, Hammarling, McKenney, Ostrouchov & Sorensen 1995), developed under the LAPACK project, for PCs, workstations, vector computers and shared memory high performance machines, which uses block-partitioned algorithms wherever possible and makes extensive use of the Level 3 BLAS, as well as the other BLAS. For background and further information on LAPACK see Dongarra & Hammarling (1990) and Dongarra, Duff, Sorensen & van der Vorst (1991). See also Dongarra, Pozo & Walker (1993) for discussion of an object oriented interface for LAPACK. At the time of writing, the development of LAPACK continues, with the Release 3 in later in 1996. Much of LAPACK has been incorporated into the NAG Fortran 77 Library and is utilized in the NAG Fortran 90 Library.

A C version of LAPACK has been produced using the f2c translator (Feldman, Gay, Maimone & Schryer 1991) and a Fortran 90 interface to LAPACK has also been proposed in Dongarra, Du Croz, Hammarling, Waśniewski & Zemla (1995).

The development of LAPACK has included the production of an extensive set of testing and timing routines. The test routines follow much the same philosophy as the development of the BLAS test programs discussed above, and are described in detail in the Installation Guide for LAPACK (Anderson, Dongarra & Ostrouchov 1992), together with a description of the timing routines. The testing of linear algebra software is discussed further in the paper by N. J. Higham in this volume.

LAPACK contain routines for the solution of dense and banded systems of linear equations, linear least squares problems and eigenvalue problems. The goals of the LAPACK project are efficiency so that the computationally intensive routines execute as fast as possible; reliability, including the return of error bounds; portability across machines;

flexibility so that users may construct new routines from well designed components; ease of use; maintainability; and good documentation.

4 SCALAPACK AND THE PBLAS AND BLACS

The ScaLAPACK project is a continuation of the LAPACK project and part of the project has been concerned with porting the LAPACK software to distributed memory parallel machines, producing the ScaLAPACK software package (Choi, Demmel, Dhillon, Dongarra, Ostrouchov, Petitet, Stanley, Walker & Whaley 1995, Choi, Dongarra, Ostrouchov, Petitet, Walker & Whaley 1995a, Choi, Dongarra & Walker 1995a). Naturally, ScaLAPACK shares the goals of LAPACK mentioned above and hence uses the same approach of promoting and utilizing standards. ScaLAPACK additionally aims at scalability as the problem size and number of processors grows on distributed memory parallel machines.

As an aid to achieving these goals the ScaLAPACK software has been designed to look as much like the LAPACK software as possible. Because the BLAS have proven to be very useful tools both within LAPACK and outside, the ScaLAPACK project chose to build a set of Parallel BLAS, or PBLAS (Choi, Dongarra, Ostrouchov, Petitet, Walker & Whaley 1995b), whose interface is as similar to the BLAS as possible. This decision has permitted the ScaLAPACK code to be quite similar, and sometimes nearly identical, to the analogous LAPACK code. Only one substantially new routine was added to the PBLAS, matrix transposition, since this is a complicated operation in a distributed memory environment (Choi, Dongarra & Walker 1995b).

It is hoped that the PBLAS will provide a distributed memory standard, just as the BLAS have provided a shared memory standard. This would simplify and encourage the development of high performance and portable parallel numerical software, as well as providing manufacturers with a small set of routines to be optimized. The acceptance of the PBLAS requires reasonable compromises among competing goals of functionality and simplicity. The PBLAS, like ScaLAPACK, perform global operations and call the BLAS to perform computations at single (local) nodes. In addition they call a set of Basic Linear Algebra Communication Subprograms, the BLACS (Dongarra & Whaley 1995), to perform the communication between processors. The BLACS can be thought of as playing the role for communication as the BLAS do for computation. The software hierarchy for ScaLAPACK is illustrated in Figure 1.

The fact that ScaLAPACK software has the same structure as LAPACK greatly facilitates the production, maintenance and portability of the software, and has also enabled the user interface to be almost the same, thus making it much easier for users to port their programs between LAPACK and ScaLAPACK. Further details of the design of ScaLAPACK, together with performance results, can be found in Choi, Demmel, Dhillon, Dongarra, Ostrouchov, Petitet, Stanley, Walker & Whaley (1995).

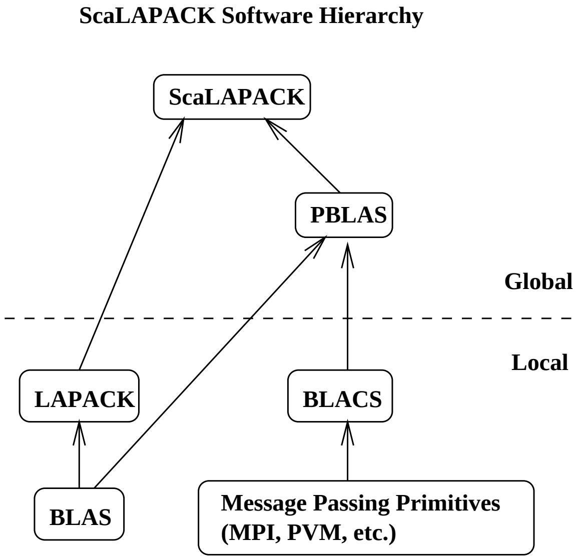

Figure 1

5 TESTING THE PBLAS

PBLAS test programs have been designed, developed and included with the PBLAS code along similar lines to those of the BLAS test programs. This test package consists of several main programs and a set of subprograms generating test data and comparing the results with those obtained by element-wise computations of the sequential BLAS. These test programs assume the correctness of the BLAS and the BLACS routines (Whaley 1995); it is therefore highly recommended that one run the testing programs provided with both of these packages before performing any PBLAS tests.

After each call to a subprogram being tested, its operation is checked in two ways. First, each of its input arguments, including all elements of the distributed operands, is checked to see if it has been altered by the subprogram. If any argument, other than the specified elements of the result scalar, vector or matrix, has been modified, an error is reported. This check includes the supposedly unreferenced elements of the distributed matrices. Second, the resulting scalar, vector or matrix computed by the subprogram is compared with the corresponding result obtained by the sequential BLAS or by simple Fortran code. We do not expect exact agreement because the two results are not necessarily computed by the same sequences of floating point operations. We do, however, expect the differences to be small relative to working precision. The error bounds are then the same as the ones used in the BLAS testers. A more detailed description of those tests can be found in Dongarra et al. (1988) and Dongarra et al. (1990).

The PBLAS testing programs are thus very similar to those for the BLAS. However, it was necessary to slightly depart from the way the BLAS testing programs operate due to the difficulties inherent in the testing of programs written for distributed-memory computers. In the following paragraphs, some essential features of the PBLAS testing programs design are presented in greater detail, together with discussion of the problems encountered, the ones we were able to solve, as well as the ones that remain open questions.

Very little distributed memory parallel programming experience is required to realize that having a program running correctly, say, on 2 processors does not necessarily imply that it will successfully run on $p > 2$ processors. Further increasing the number of potential test cases is the fact that parallel dense linear algebra kernels ordinarily assume a processor grid, typically a two dimensional grid. Furthermore, a general software library such as the PBLAS has to behave correctly even in degenerate cases, such as when the distributed matrix does not span all processors in one or both dimensions of the grid. Finally, it should also be possible to vary the size and location of the submatrices to operate on, the data decomposition parameters such as the block sizes used for the matrix partitioning and distribution, or even the local leading dimension of the arrays that locally store the pieces of the distributed matrices. Note that none of these remarks apply to the sequential testing problem.

These remarks suggest that it is in practice impossible to test even a very small portion of all the possible different test cases. However, it is important to be able to generate any possible case, so that the tester can also be used to check a given operation for a particular data distribution.

These facts motivated the decision to permit a user configurable set of tests for every PBLAS routine. Concretely, the input testing files allow for the precise specification of a limited number of tests. The input files for each test contain for each test, a complete description of the data layout of each operand allowing one to mimic exactly a given call to a PBLAS subroutine. Consequently, one can test the PBLAS with any possible machine configuration as well as data layout. The obvious drawback of such generality is that the input testing file is slightly longer and more complex than the input files used for the sequential BLAS testers.

The PBLAS software follows an SPMD or data-parallel programming model. If a PBLAS routine is called with an invalid value for any of its arguments, then it must report the fact and terminate the execution of the program. In the model implementation, each routine, on detecting an error, calls a common error-handling routine.

This input error checking aspect of the software is also tested. It is straightforward to plug in an erroneous combination of input arguments and check that the error handler behaves correctly. It is however interesting to notice that a PBLAS routine cannot ensure that every process does indeed call this subroutine.

Since checking arguments in a global fashion would add a global synchronization step, for efficiency purposes, the PBLAS routines only perform a local validity check of their argument list. If a value is invalid in at least one process of the current context, the program execution is stopped. As a result, different processes may have different values of an argument that should be the same, thus causing non predictable results. . We comment further on the problems of networks of heterogeneous computers in Section 8 below.

6 TESTING SCALAPACK

The ScaLAPACK test programs are modeled on those of LAPACK, but are not yet as extensive as those of LAPACK. As well as the additional complexity mentioned above, ScaLAPACK testing can put a serious strain on memory requirements since testing generally requires one or more additional copies of matrices to compute quantities such as the residuals required for the error bounds. Currently the testing routines also serve as the timing routines, but it is hope to change that in the future since the additional memory requirements can severely limit the size of matrix.

Further details on testing and timing for ScaLAPACK can be found in Choi, Dongarra, Ostrouchov, Petitet & Whaley (1995).

7 THE NAG NUMERICAL PVM LIBRARY

The NAG Library does not readily port to distributed memory machines, and many of the routines in the Library are either not designed for parallel machines, or are performing functions that are not appropriate in a parallel setting. Even if the Library were suitable, the software support is such that we could not currently contemplate such a port. We hope that in the future HPF (Koelbel, Loveman, Schreiber, Steele Jr. & Zosel 1994) may prove to be a suitable vehicle, but the language and the compilers are not yet sufficiently mature for us to seriously contemplate the use of HPF.

Following the ScaLAPACK lead NAG therefore took the somewhat reluctant step of developing a parallel message passing library, based initially upon PVM (Geist, Beguelin, Dongarra, Jiang, Manchek & Sunderam 1994). This library has been carefully designed with the future very much in mind, both short term and longer term. In the short term NAG are, at the time of writing, about to distribute an MPI (Snir, Otto, Huss-Lederman, Walker & Dongarra 1996) version of the library, since MPI is set to become widely available as the (*de facto*) standard message passing system. In the longer term the hope is that this library, together with the Fortran 90 Library, can feed into an HPF library activity. Thus, as with ScaLAPACK, NAG has adopted an SPMD model of parallelism for the library, in which (possibly) many instances of a single program are executed concurrently on different data sets; NAG has tried to limit the use of PVM as much as possible, using the BLACS wherever it is sensible to do so; and, in common with ScaLAPACK, assumes a two-dimensional logical grid.

The use of the SPMD model is not always straightforward, for example in the quadrature algorithms a farming model is more natural, but it was nevertheless felt sensible to adhere to the SPMD model since that is the current HPF model of programming, and it does not really place any undue restriction on the user. The user may spawn off other tasks so long as the group of library tasks behave according to the SPMD model. NAG are also providing utility routines to further insulate the user and themselves from the underlying message passing system. Before calling the library the user calls an initialization routine, and calls an exit routine at the end, along the lines of the following example.

```
*     .. call the NAG initialization routine ..
         call nagspawn( ... )
         :
         call d02hafp( ... )
         :
         call f07adf( ... )
         :
*     .. call the NAG clean-up routine ..
         call nagexit( ... )
```

Initializing PVM and/or the BLACS (or any other future communication system that is adopted) is done within `nagspawn`. From the user's perspective the error handling mechanism and interface is the same as for the Fortran 77 Library, with some additional error checks such as ensuring that a global variable has the same value on all processors.

Naturally NAG have adapted their stringent test programs, in a similar manner to ScaLAPACK, to the distributed memory environment in order to maintain their reputation for numerical quality and reliability.

8 HETEROGENEOUS COMPUTING ENVIRONMENTS

In principal, both ScaLAPACK and the NAG parallel library can be run on networks of heterogeneous machines, but in this final section we mention the special challenges associated with writing and testing numerical software that is to be executed on networks containing heterogeneous processors, that is, processors which perform floating point arithmetic differently. This includes not just machines with different floating point formats and semantics such as Cray vector computers and workstations performing IEEE standard floating point arithmetic, but even supposedly identical machines running different compilers, or even just different compiler options or runtime environments.

Moreover, on such networks, floating point data transfers between two processes may require a data conversion phase and thus a possible loss of accuracy. It is therefore impractical, error-prone and difficult to compare supposedly identical computed values on such heterogeneous networks. As a consequence, the validity and correctness of the tests performed can only currently be guaranteed for networks of processors with identical floating point formats.

It is not enough to require identical floating point representation across all processors of a parallel computer. The way arithmetic is performed should also agree to some extent. For example, having a processor in the network that does not produce and recognize denormalized number representations can cause problems when receiving such a number from other processors that can properly generate denormalized numbers. We have not yet tried to make the testing programs generate input data on the edge of the floating point number range, in order to identify and trap these problems. Whilst this is highly desirable, we have not yet sufficiently investigated the generation of such test problems to be confident of exposing the difficulties.

Further discussion of the dangers of heterogeneous computing can be found in Demmel, Dongarra, Hammarling, Ostrouchov & Stanley (1996) and citeasnounBCDDDHPRSW:UTK-cs:96.

9 FURTHER INFORMATION

Many working notes have been produced as part of the LAPACK and ScaLAPACK projects and these are available from `http://www.netlib.org/lapack/lawns/`

10 ACKNOWLEDGMENTS

We wish to thank our ScaLAPACK colleagues at the University of Tennessee at Knoxville and the University of California at Berkeley, as well as colleagues at NAG involved with the development of parallel software, for their valuable input.

REFERENCES

Anderson, E., Bai, Z., Bischof, C. H., Demmel, J., Dongarra, J. J., Du Croz, J., Greenbaum, A., Hammarling, S., McKenney, A., Ostrouchov, S. & Sorensen, D. C. (1995). *LAPACK Users' Guide*, 2nd edn, SIAM, Philadelphia, PA, USA. (Also available in Japanese, published by Maruzen, Tokyo, translated by Dr Oguni).

Anderson, E., Dongarra, J. J. & Ostrouchov, S. (1992). Installation guide for LAPACK. LAPACK Working Note No.41, *Technical Report CS-92-151*, Department of Computer Science, University of Tennessee, 107 Ayres Hall, Knoxville, TN 37996-1301, USA.

Choi, J., Demmel, J., Dhillon, I., Dongarra, J. J., Ostrouchov, S., Petitet, A., Stanley, K., Walker, D. W. & Whaley, R. C. (1995). ScaLAPACK: A portable linear algebra library for distributed memory computers - design issues and performance, *in* J. J. Dongarra, K. Masden & J. Waśniewski (eds), *Applied Parallel Computing*, Springer-Verlag, Berlin, Germany, pp. 95–106. (Proceedings of the Second International Workshop, PARA '95, Lyngby, Denmark. See also LAPACK Working Note No.95).

Choi, J., Dongarra, J. J., Ostrouchov, S., Petitet, A., Walker, D. W. & Whaley, R. C. (1995a). The design and implementation of the reduction routines in ScaLAPACK, *in* J. J. Dongarra, L. Grandinetti, G. R. Joubert & J. Kowalik (eds), *High Performance Computing: Technology, Methods and Applications*, Advances in Parallel Computing, 10, Elsevier, Amsterdam, The Netherlands, pp. 177–202. (See also LAPACK Working Note No.80).

Choi, J., Dongarra, J. J., Ostrouchov, S., Petitet, A., Walker, D. W. & Whaley, R. C. (1995b). A proposal for a set of parallel basic linear algebra subprograms, *in* J. J. Dongarra, K. Masden & J. Waśniewski (eds), *Applied Parallel Computing*, Springer-Verlag, Berlin, Germany, pp. 107–114. (Proceedings of the Second International Workshop, PARA '95, Lyngby, Denmark. See also LAPACK Working Note No.100).

Choi, J., Dongarra, J. J., Ostrouchov, S., Petitet, A. & Whaley, R. C. (1995). Installation guide for ScaLAPACK. LAPACK Working Note No.93, *Technical report*, Department of Computer Science, University of Tennessee, 107 Ayres Hall, Knoxville, TN 37996-1301, USA.

Choi, J., Dongarra, J. J. & Walker, D. W. (1995a). The design of a parallel dense linear algebra software library: Reduction to Hessenberg, tridiagonal and bidiagonal form, *Numerical Algorithms* **10**: 379–399. (See also LAPACK Working Note No.92).

Choi, J., Dongarra, J. J. & Walker, D. W. (1995b). Parallel matrix transpose algorithms on distributed memory concurrent computers, *Parallel Computing* **21**: 1387–1405.

Demmel, J., Dongarra, J. J., Hammarling, S., Ostrouchov, S. & Stanley, K. (1996). The dangers of heterogeneous network computing: Heterogenous networks considered harmful, *Proceedings Heterogeneous Computing Workshop '96*, IEEE Computer Society Press, Los Alamitos, CA, USA, pp. 64–71.

Dodson, D. S. (1983). Corrigendum: Remark on "Algorithm 539: Basic Linear Algebra Subroutines for FORTRAN usage", *ACM Trans. Math. Software* **9**: 140. (See also (Lawson et al. 1979) and (Dodson & Grimes 1982)).

Dodson, D. S. & Grimes, R. G. (1982). Remark on algorithm 539: Basic Linear Algebra Subprograms for Fortran usage, *ACM Trans. Math. Software* **8**: 403–404. (See also (Lawson et al. 1979) and (Dodson 1983)).

Dongarra, J. J., Bunch, J. R., Moler, C. B. & Stewart, G. W. (1978). *LINPACK Users' Guide*, SIAM, Philadelphia, PA, USA.

Dongarra, J. J., Du Croz, J., Duff, I. S. & Hammarling, S. (1990). A set of Level 3 Basic Linear Algebra Subprograms, *ACM Trans. Math. Software* **16**: 1–28. (Algorithm 679).

Dongarra, J. J., Du Croz, J., Hammarling, S. & Hanson, R. J. (1988). An extended set of FORTRAN Basic Linear Algebra Subprograms, *ACM Trans. Math. Software* **14**: 1–32, 399. (Algorithm 656).

Dongarra, J. J., Du Croz, J., Hammarling, S., Waśniewski, J. & Zemla, A. (1995). A proposal for a Fortran 90 interface for LAPACK, *in* J. J. Dongarra, K. Masden & J. Waśniewski (eds), *Applied Parallel Computing*, Springer-Verlag, Berlin, Germany, pp. 158–165. (Proceedings of the Second International Workshop, PARA '95, Lyngby, Denmark. See also LAPACK Working Note No.101).

Dongarra, J. J., Duff, I. S., Sorensen, D. C. & van der Vorst, H. A. (1991). *Solving Linear Systems on Vector and Shared Memory Computers*, SIAM, Philadelphia, PA, USA.

Dongarra, J. J. & Grosse, E. (1987). Distribution of mathematical software via electronic mail, *Communs Ass. comput. Mach.* **30**: 403–407.

Dongarra, J. J. & Hammarling, S. (1990). Evolution of numerical software for dense linear algebra, *in* M. G. Cox & S. Hammarling (eds), *Reliable Numerical Computation*, Oxford University Press, Oxford, UK, pp. 297–327.

Dongarra, J. J., Pozo, R. & Walker, D. W. (1993). An object oriented design for high performance linear algebra on distributed memory architectures, *Proceedings of OON-*

SKI '93: First Object-Oriented Numerics Conference, Rogue Wave Software, Corvallis, OR, USA, pp. 257–264.

Dongarra, J. J., Rowan, T. H. & Wade, R. C. (1993). Software distribution using Xnetlib, *Technical Memorandum ORNL/TM-12318*, Oak Ridge National Laboratory, Oak Ridge, Tennessee 37831, USA.

Dongarra, J. J. & Whaley, R. C. (1995). A users' guide to the BLACS v1.0. LAPACK Working Note No.94, *Technical Report CS-95-281*, Department of Computer Science, University of Tennessee, 107 Ayres Hall, Knoxville, TN 37996-1301, USA.

Feldman, S. I., Gay, D. M., Maimone, M. W. & Schryer, N. L. (1991). A Fortran-to-C converter, *Computer Science Technical Report 49*, AT&T Bell Laboratories, Murray Hill, NJ 07974, US.

Geist, A., Beguelin, A., Dongarra, J. J., Jiang, W., Manchek, R. & Sunderam, V. (1994). *PVM: Parallel Virtual Machine. A Users' Guide and Tutorial for Networked Parallel Computing*, MIT Press, Cambridge, MA, USA.

Hammarling, S. (1993). Development of numerical software libraries for vector and parallel machines, *in* A. E. Fincham & B. Ford (eds), *Parallel Computation*, Oxford University Press, Oxford, UK, pp. 11–35.

Koelbel, C. H., Loveman, D. B., Schreiber, R. S., Steele Jr., G. L. & Zosel, M. E. (1994). *The High Performance Fortran Handbook*, The MIT Press, Cambridge, MA, USA.

Lawson, C. L., Hanson, R. J., Kincaid, D. & Krogh, F. T. (1979). Basic Linear Algebra Subprograms for FORTRAN usage, *ACM Trans. Math. Software* **5**: 308–323. (Algorithm 539. See also (Dodson & Grimes 1982) and (Dodson 1983)).

Snir, M., Otto, S. W., Huss-Lederman, S., Walker, D. W. & Dongarra, J. J. (1996). *MPI: The Complete Reference*, MIT Press, Cambridge, MA, USA.

Whaley, R. C. (1995). Installing and testing the BLACS, *Technical report*, Department of Computer Science, University of Tennessee, 107 Ayres Hall, Knoxville, TN 37996-1301, USA. (See http://www.netlib.org/blacs/BLACS/Install.html).