# Background, Motivation, and a Retrospective View of the BLAS

Charles L. Lawson
Retired from Jet Propulsion Laboratory
California Institute of Technology
Home: 301 Calle Pueblo
San Clemente, CA 92672
clawson@na-net.ornl.gov

## 1 Introduction

During the initial development of computers in the 1940's and early 1950's the chain of intermediate technologists between the computer designer/builder and the computer user was very short. In fact it was not uncommon for one individual to be actively involved in the design, construction, and application of a computer. By 1970 there was a significant amount of infrastructure between the computing machine and the end user - for example, from low-level to high-level one could cite operating systems, compilers, general-purpose software, and applications-specific software. Each level of this infrastructure absorbed the creativity and energies of its own community of specialists trying to improve its link in the chain. This frequently took the form of trying to better meet the perceived needs of the higher levels in the infrastructure while coping with the limitations and peculiarities of the resources provided by the lower levels.

The original BLAS [16] constituted a contribution to the general-purpose mathematical software stratum of the computing infrastructure. To better understand the influences that led to their development, let us very briefly recount some of the activity regarding general-purpose mathematical software in the early 70's.

## 2 Background and Motivation for the BLAS

Through the late 1960's and up to about 1971 a number of collections or libraries of mathematical software had been assembled. Typically each of these libraries was developed for, and useable on, just one brand of computer system. Examples would be a library collected and/or developed by a computer vendor for use with their computer, or a library collected and/or developed by a large organization's computing center for use within that organization. For instance, Michael Powell recalls developing a math library at the Harwell Laboratory in the early 1960's.

Even the two organizations, NAG [5] and IMSL [12], that later became the world's main suppliers of math libraries across diverse machine types, each started in 1970 by targeting a single machine type. NAG started as an informal cooperative effort by six institutions in England to develop a math library for their newly acquired ICL 1906A systems. Their first library, in both Algol and Fortran versions, was available to the participating institutions in October, 1971, and they immediately started attacking the problem of developing the library for other computer types.

IMSL was founded in 1970 as a company to develop and market mathematical and statistical libraries. A major motivation for the founding of IMSL was a perception of customer dissatisfaction with the math libraries provided by IBM for use on their systems. IMSL's first product, available in 1971, was a Fortran library for the IBM/370-360.

Designing math library software to be portable across diverse systems was particularly difficult in the 70's because there were significant differences between the arithmetic characteristics of computers from different manufacturers, the state of standardization of programming languages was in its infancy, and operating systems (encompassing file naming and file storage) were quite different on different computer brands. Fortran was the only language widely supported in the U.S., and the only language for which there was an ANSI standard (1966). However the standardized

language had major shortcomings so programmers tended to use vendor-specific extensions and that worked against portability. Algol and Fortran were both used in Europe.

A project called NATS (National Activity for Testing Software or NSF, Argonne, Texas, Stanford) [1] undertook during 1971-1972 to investigate the problems of achieving Fortran portability for mathematical software by testing, and as necessary modifying, a Fortran version of a set of dense matrix eigenvalue/eigenvector algorithms. These 30 algorithms were originally developed in Algol and published in a series of papers in the 1960's by J. H. Wilkinson and others, and subsequently published together as the *Handbook* ... [18]. The collection was converted to Fortran by Virginia Klema and others at Argonne National Laboratory and then processed for portability by NATS. NATS announced the availability of EISPACK [2, 17, 6] in 1972, the package having been successfully tested on IBM 360-370, CDC 6000-7000, Univac 1108, Honeywell 635, and PDP-10 computers. The same EISPACK code ran on all these systems, with the exception of one variable that needed to be set to indicate the precision of the arithmetic on the host system.

EISPACK was widely requested and distributed. This project demonstrated the great utility of carefully tested portable mathematical software based on state of the art algorithmic research. But did (Fortran) portability mean sacrificing efficiency? Library developers began giving more attention to this issue, and new hardware designs presented new challenges and opportunities. For example the CDC 7600 had an instruction cache of about 16 instructions. If a loop were complete in this number of instructions it would execute much faster than would a longer instruction sequence.

Experience with library usage showed that linear algebra codes were the most heavily used, and in these codes most of the execution time is spent in inner loops that implement either a dot product of two vectors, or the “elementary vector operation" of replacing a vector, **y**, by **y** plus the product of a scalar, *a*, and a vector, **x**. Thus improving the efficiency of these operations in portable software could be expected to have a worthwhile payoff in improving the efficiency of scientific computing generally.

## 3 Developing the BLAS

At the Jet Propulsion Laboratory (JPL), Richard Hanson, Fred Krogh, and Charles Lawson were in a numerical mathematics group that, among other tasks, developed and supported an in-house math library for the Univac 1108, from about 1969 to 1977 (and in subsequent years developed libraries portable in Fortran 77 and ANSI C, known respectively as MATH77 and Mathc90.) A 1972 internal memo by Krogh [13] reported on a timing study comparing code written in Fortran in two different ways, and written in Univac assembly code, for the two basic vector operations noted above, plus the Euclidean norm and finding the element of largest magnitude in a vector.

Believing that developers of library-type math software could benefit from the availability of efficient portable software modules for these basic vector operations, H., K., & L. came up with a proposal [10] in November, 1973 for such a collection as a candidate for widespread use. David Kincaid, University of Texas, became actively involved, giving particular attention to assembly coded versions for the CDC 6600.

The ACM SIGNUM Newsletter was used as a vehicle to communicate this proposal and its subsequent evolution [9, 14]. (Recall that we did not have the Internet and email in those years, and the SIGNUM Newsletter was a very useful vehicle for the informal exchange of information in the numerical analysis community with less delay than a refereed journal.) A lively open meeting was held to discuss and modify the proposal in May, 1974, at the Math Software II Conference at Purdue University, with another less eventful meeting at the National Computer Conference in Anaheim, May, 1975. By 1977, Hanson was at Sandia, Albuquerque, and the Sandia report [11] issued in 1977 was essentially the final version. This final version appeared in ACM TOMS in September, 1979 [16].

The LINPACK project was ongoing during this same time period, having started shortly after the first release of EISPACK, and publishing LINPACK [4] in 1979. The LINPACK project made an early decision to use the BLAS. Jack Dongarra, who was a coauthor of LINPACK made

substantial contributions, [3], to the testing and coding of the BLAS, particularly adding loop-unrolling to the Fortran versions.

## 4 What operations are in the BLAS?

The choice of functionalities included in the BLAS was based on the goal of supporting the writing of library routines for the dense matrix problems of linear equation solving, eigensystems, and linear least-squares. It was agreed to include only operations that involved a single level of looping. Following is a table of the functionalities provided, along with the root name for each functionality, and the set of prefixes, or prefix/suffix pairs, allowed with each root name.

| | | |
|---|---|---|
| Dot product | s, ds, sds, d, dq_ _i, dq_ _a, c_ _c, c_ _u | dot |
| Scalar times a vector plus a vector | s, d, c | axpy |
| Construct Givens plane rotation | s, d | rotg |
| Apply a plane rotation | s, d | rot |
| Construct a modified Givens rotation | s, d | rotmg |
| Apply a modified rotation | s, d | rotm |
| Copy a vector | s, d, c | copy |
| Swap two vectors | s, d, c | swap |
| Two-norm of a vector | s, d, sc | nrm2 |
| Sum of magnitudes of vector components | s, d, sc | asum |
| Scalar times a vector | s, d, c, cs | scal |
| Index of element of largest magnitude | is, id, ic | amax |

The letters s, d, c, q, and i in prefixes denote respectively single precision, double precision, single precision complex, extended precision, and integer data types. The letters c, u, i, and a in suffixes denote respectively conjugated, unconjugated, initial, and accumulating.

There are a total of 38 subprograms. The original 1973 proposal, [10], had 39 subprograms. Of these, two had a modal argument, and in the final package each of these subprograms was replaced by two subprograms to avoid having modal arguments. The subprogram sdsdot was added and four subprograms from the original proposal were dropped, leaving the number of subprograms in the final package one less than in the original proposal.

Some of the root names were changed, particularly at the 1974 Purdue meeting. The root name axpy denoting "*a* times x plus y", replaced the original name elvop which denoted "elementary vector operation". The names sdot and saxpy (or ddot and daxpy) have become standard terminology in the literature of computational linear algebra to denote these fundamental operations. The name saxpy was adopted as the name of a company manufacturing a workstation in the 1980's, presumably to remind the marketplace that it was designed to do saxpy's rapidly.

Double precision complex data types were not included in the BLAS package because this data type was not included in the Fortran 66 or Fortran 77 standards. The LINPACK project did include this data type however, and it is suggested in [16] that the letter z be used in names for this data type, as was done in LINPACK, if the package is extended to handle it.

A fair amount of publicity was given in the 1960's to the accuracy advantages of using extra precision in the accumulation of inner products. This probably led to our inclusion of the various mixed precision dot product subprograms. The now widely used IEEE arithmetic units, that carry a few extra bits of precision internally, make these mixed precision subprograms less important on contemporary computers.

Computation of the two-norm of a vector is well known to present the hazard that intermediate results may need twice the exponent range of the input numbers and of the final result. Again the IEEE arithmetic units give some relief for this, as they carry extra bits in the exponent internally. The two-norm subprograms in the BLAS use an algorithm devised for the BLAS by Lawson that does the computation with just one pass through the vector, doing scaling on the fly if necessary to avoid overflow or underflow if it is avoidable and would affect the accuracy of the result.

A modified algorithm for the generation and application of Givens rotations (of use, e.g., in least-squares computations) that used two multiplies and two adds, instead of four multiplies and two adds, per pair of elements transformed, was devised in the early 1970's by Gentleman [7] and by Hammarling [8]. This modification has scaling hazards and inherent complexity that can easily discourage one from trying to implement it, and the potential reduction of execution time is fairly modest. We attempted with the modified Givens subprograms in the BLAS to encapsulate the complexity and handle the scaling in a way that would make it more straightforward for persons to experiment with the use of this method. Specifically we implemented the approach described in [15].

## Conclusions

One of the motivations for developing the BLAS was to provide names and argument lists that might become widely used and recognized for some of the basic operations of computational linear algebra. It was hoped that this would aid software developers by providing a standard set of building blocks, and also aid the person who is faced with the need to understand code written by someone else. I think the original BLAS have served this role quite well, having been used in many software development projects.

Another motivation was to improve the efficiency of math software, by providing standardized subprogram interfaces with the possibility of improving the efficiency of the code beyond the interface to take advantage of special machine features. This potential was realized with the provision of machine tuned versions of the BLAS on numerous brands of computers, especially providing efficient implementations of `_dot` and `_axpy`.

With the development of vector machines in the late 1970's and numerous variations of pipelining, cache, multilevel memory systems, and parallel processing systems in the 1980's and 1990's, it was observed that the vector operations of the original BLAS package were too limited to exploit the possible efficiencies of these new machine architectures. The model of a BLAS type package still appeared to be useful but it was seen to be necessary to encapsulate matrix-vector and matrix-matrix operations as well as the vector operations of the original BLAS, and design specifically for parallel processing. This led to the successor BLAS packages that are the subject of the other speakers in this minisymposium.

## References


[1] ANNOUNCEMENT, *NATS Project - Collaborative Research toward the Development of a Certified Sub-routine Library*, ACM SIGNUM Newsletter, 6, 3, Nov 1971, p. 5.

[2] ANNOUNCEMENT, *The Certified Eigensystem Package*, EISPACK, ACM SIGNUM Newsletter, 7, 2, July 1972, pp. 4-5.

[3] J. J. DONGARRA, *Fortran BLAS Timing - LINPACK Working Note 3*, Argonne National Laboratory, Argonne, IL, draft of March 1977.

[4] J. J. DONGARRA, J. R. BUNCH, C. B. MOLER, AND G. W. STEWART, *LINPACK Users Guide*, SIAM, Philadelphia, PA, 1979.

[5] B. FORD, *The Nottingham Algorithms Group (NAG) Project*, ACM SIGNUM Newsletter, 8, 2, April 1973, pp. 16-21.

[6] B. S. GARBOW, J. M. BOYLE, J. J. DONGARRA, AND C. B. MOLER, *Matrix Eigensystem Routines - EISPACK Guide Extension*, Vol. 51 of Lecture Notes in Comput. Sci., Springer-Verlag, Berlin, 1977.

[7] W. M. GENTLEMAN, *Least Squares Computations by Givens Transformations without Square Roots*, J. Inst. Math. Appl., 12, 1973, 329-336.

[8] S. J. HAMMARLING, *A Note on Modifications to the Givens Plane Rotation*, J. Inst. Math. Appl., 13, 1974, 215-218.

[9] R. J. HANSON, F. T. KROGH, AND C. L. LAWSON, *Improving the Efficiency of Portable Software for Linear Algebra*, ACM SIGNUM Newsletter, 8, 4, Oct 1973, p. 16.

[10] R. J. HANSON, F. T. KROGH, AND C. L. LAWSON, *A Proposal for Standard Linear Algebra Subprograms*, Jet Propulsion Laboratory Technical Memorandum 33-660, November, 1973, vi+14 pp.

[11] R. J. HANSON, C. L. LAWSON, D. R. KINCAID, AND F. T. KROGH, *Basic Linear Algebra Subprograms for FORTRAN Usage - An extended report*, Sandia Tech. Rep. SAND 77-0898, Sandia Lab., Albuquerque, NM, 1977.

[12] O. G. JOHNSON, *IMSL's ideas on subroutine library problems*, ACM SIGNUM Newsletter, 6, 3, Nov 1971, pp. 10-12.

[13] F. T. KROGH, *On the Use of Assembly Code for Heavily Used Modules in Linear Algebra*, JPL Internal Memorandum, May 2, 1972, 13 pp.

[14] C. L. LAWSON, *Proposed Standard Subprograms for Basic Linear Algebraic Operations*., ACM SIGNUM Newsletter, 9, 2, April 1974, pp. 21-22.

[15] C. L. LAWSON, AND R. J. HANSON, *Solving Least Squares Problems*, Prentice-Hall, 1974. Republished with an updating appendix by SIAM, 1996.

[16] C. L. LAWSON, R. J. HANSON, D. R. KINCAID, AND F. T. KROGH, *Basic Linear Algebra Subprograms for Fortran Usage*, ACM TOMS, 5, 3, September, 1979, pp. 308-323, and Algorithm 539, pp. 324-325.

[17] B. T. SMITH, J. M. BOYLE, B. S. GARBOW, Y. IKEBE, V. C. KLEMA, AND C. B. MOLER, *Matrix Eigensystem Routines - EISPACK Guide*, Vol. 6 of Lecture Notes in Comput. Sci., Springer-Verlag, Berlin, 1974 (2nd ed., 1976, with additional author, J. J. DONGARRA).

[18] J. H. WILKINSON AND C. REINSCH, eds., *Handbook for Automatic Computation, Vol. 2, Linear Algebra*, Springer-Verlag, New York, 1971.