The Influence and Contribution of Jack Dongarra to Numerical Linear Algebra

Sven Hammarling and Nicholas J. Higham, *The University of Manchester, Manchester, M13 9PL, U.K.*

This article is dedicated to Jack Dongarra on the occasion of him receiving the 2021 ACM Turing Award and concentrates primarily on his contributions to numerical linear algebra, particularly on the development of algorithms and software to reliably and efficiently solve linear algebra problems. We shall look at software projects, a number of them initiated by Jack, ranging from the BLAS and LINPACK through LAPACK to recent projects, such as SLATE, as well as some of his algorithmic contributions. Quoting from one of Jack's own recent talks, his tenets have been "accuracy, community, innovation, performance, portability, productivity, readability, and reliability."

For four decades, Jack Dongarra has had a profound impact on the design and development of algorithms, software, and underpinning programming standards for high-performance computing (HPC), especially in the area of linear algebra. His algorithms and open-source software libraries are used throughout scientific computation and will be crucial over the coming years in order to allow scientific applications to exploit computing systems running at exascale and beyond. His software involves the innovative exploitation of memory hierarchies, intelligent performance tuning, and the use of novel numerical algorithms and comprehensive error bounds to achieve performance and portability.

On the occasion of him receiving the 2021 ACM A.M. Turing Award, we give a high-level overview of Dongarra's contributions to numerical linear algebra. Further details can be found in the references.

EARLY DAYS

The first author, Sven, first met Jack in 1975 or 1976 when Sven was on sabbatical leave at the National Physical Laboratory, working for Jim Wilkinson.[a] Jack was going to the U.K. and his future Ph.D. supervisor, Cleve Moler, suggested that he visit Jim. Jim was a big influence on the careers of Jack and Sven and is a hero to us both, as well as to Nick. The meeting between Jack and Sven led to a lifetime friendship and a number of collaborative projects. Jack was working at the Argonne National Laboratory at the time. He started as a resident student associate in 1973 and left as a Senior Computer Scientist in 1989 for the University of Tennessee. Wilkinson was a regular visitor to Argonne, as was Cleve Moler, and we are sure that Jack would be the first to say that they both had a big impact on him.

Jack invited me (Sven) to Argonne in the summer of 1989, but he and his colleagues, Brian Smith and Danny Sorenson, all left for new pastures during my visit!! I wondered what I had done!

Nick first met Jack in 1984 at the Gatlinburg meeting on Numerical Linear Algebra in Waterloo, Canada. In 2007, Jack was appointed to a part-time position of professor and Turing Fellow at the University of Manchester, and has spent most summers in Manchester since then. In light of his Turing Award, it seems to be a particularly appropriate title!

EISPACK, LINPACK, BASIC LINEAR ALGEBRA SUBPROGRAMS (BLAS), AND PH.D.

In the 1970s, before Jack completed his Ph.D., three projects important to Jack's career were completed, two of them under the auspices of Argonne. EISPACK, the first to be completed in 1974, was a collection of Fortran routines for solving matrix eigenvalue problems. EISPACK

[a]The recipient of the Turing Award in 1970. For further information about Wilkinson see the authors' web page at https://nla-group.org/james-hardy-wilkinson/

1521-9615 © 2022 IEEE
Digital Object Identifier 10.1109/MCSE.2022.3198330
Date of publication 15 August 2022; date of current version 4 January 2023.

was based on the Algol eigenvalue routines from the landmark linear algebra book *The Handbook* edited by Wilkinson and Reinsch, published in 1971.[b] *The Handbook* set the standard for portable linear algebra software, not just for the routines, but also the documentation, including background, applicability, description of parameters, and discussion of the numerical properties, error bounds, and test results. It certainly influenced Jack as well as the two present authors. Sven, though, is the only one of us to have programmed in Algol! Jack was not an author of the first edition of EISPACK,[1] but he was put to work on testing the routines and he was an author of the two further editions, published in 1976 and 1977.

The second project, for which Jack was an author from the outset, was LINPACK,[2] a package for solving linear equations and least squares problems. While the algorithms used were strongly influenced by Jim Wilkinson and *The Handbook*, it was not a translation of *The Handbook* routines into Fortran. Since Algol stored arrays by row and Fortran by column, it was realized that a straight translation could lead to serious cache misses, so the code accessed the arrays by column wherever possible.

Jack's first published paper, joint with an Argonne colleague A. R. Hinds, was on unrolling loops in Fortran,[3] the start of Jack's concern with performance. The technique was demonstrated on routines for dot products, such as $d = x^T y$, and the so-called axpy operation $y = ax + y$, where x and y are vectors and a is a scalar. Essentially, the idea was to pipeline the operations so that, for example, instead of performing just $x_i y_i$ in a loop, several products are performed in a loop, giving each loop more arithmetic.

The two routines used in this article were part of the BLAS,[c] which was a community effort to define a standard interface for routines to perform basic linear algebra operations. An important aim was to allow tuned versions of the routines, but without the user having to change the calling routine. Jack, although not an author, did have a strong acknowledgment both for his testing and for his work on implementations.[4]

We can see how Jack was influenced and enthused by his early experiences into a lifelong belief about the tenets "accuracy, community, innovation, performance, portability, productivity, readability, and reliability," that he mentioned in a recent talk.[d] We believe collaboration and standards should be added to those tenets.

What about his contribution to algorithms? Jack did do some algorithmic work for his master's thesis, reducing large banded matrices to tridiagonal form under the supervision of Brian Smith, who Jack credits with being instrumental in getting him involved with numerical methods and mathematical software.[e] It also gave him the experience of optimizing memory movement. That was in 1973. His Ph.D. thesis was completed in 1980 and was concerned with a method for "Improving the Accuracy of Computed Matrix Eigenvalues." Wilkinson had given a talk about the subject at Argonne, which Cleve Moler thought would be ideal for Jack's thesis. The thesis appeared as an Argonne Technical Report[f] and subsequently as a joint paper by Dongarra, Moler, and Wilkinson.[5] Naturally, Jack also produced software, SICEDR, to implement and test the method, which is still available from the ACM ToMS collection[g]. Soon afterward Jack extended the ideas to singular values.[6]

Something of Jack's early years and much about his later life up to about 2004 can be found in an interview[h] and a biography,[i] both carried out by Thomas Haigh as part of the SIAM History Project.

LEVEL 2 AND 3 BLAS, LAPACK, AND SCALAPACK

As computers became more sophisticated, Jack and others realized that, for algorithms of numerical linear algebra, one needed to optimize operations at a higher level than the Level 1 BLAS vector operations. For instance, on vector machines, we needed to optimize at least at the level of matrix–vector operations. This requirement led to the development of the Level 2 BLAS by Dongarra, Du Croz, Hammarling, and Hanson,[7] for which Jack brought in Jeremy Du Croz (who was also working on a basic set of matrix–vector operations) and Sven, from the Numerical Algorithms Group (NAG).

More or less concurrently with the publication of the Level 2 BLAS it was realized that machines with a hierarchy of memory, or true parallelism, needed algorithms that partitioned matrices into blocks and performed matrix–matrix operations on the blocks, hence the development of the Level 3 BLAS, by Dongarra, Du Croz, Duff, and Hammarling.[8] The Level 2 and 3 BLAS

[b]Handbook for Automatic Computation, Volume II, Linear Algebra, Springer-Verlag.
[c]Now known as the Level 1 BLAS.
[d]"A Not So Simple Matter of Software," E-NLA seminar, April 13, 2022, https://youtu.be/cgP1VqHWxpo.

[e]Jack regards Brian Smith, Jim Wilkinson, Cleve Moler, Gene Golub, and Ken Kennedy as his mentors.
[f]ANL-80-84.
[g]File 589.gz at https://calgo.acm.org/.
[h]http://history.siam.org/oralhistories/dongarra.htm
[i]http://www.netlib.org/utk/people/JackDongarra/PAPERS/DongarraBio.pdf

projects were led by Jack and developed in the same spirit as the Level 1 BLAS, with the involvement of and consultation with the community.

EISPACK and LINPACK were not expressed in terms of Level 2 or 3 BLAS. This provided the impetus for the LAPACK project, led by Dongarra and James Demmel (Berkeley), which produced a new library having the capabilities of EISPACK and LINPACK and much more besides. LAPACK employs block-partitioned algorithms to ensure high levels of data reuse; these include both new algorithms (such as the divide and conquer eigenvalue algorithm) as well as innovations that allowed existing algorithms to be recast in terms of Level 2 and 3 BLAS.

The first public release of LAPACK was on February 29th, 1992, on Gene Golub's 15th leap day birthday, or his 60th birthday, when a first edition of the manual was presented to him. The royalties from the sales of the manual were given to the SIAM Student Travel fund. LAPACK was intended to supersede EISPACK and LINPACK as well as to add functionality, such as condition number estimates, error bounds for linear equations, and iterative refinement. There were two further releases of the manual in 1995[j] and 1999.[9] From then on releases were documented on the LAPACK website,[k] the latest update at the time of writing being 3.10.1, dated April 12, 2022, just over 30 years since the first release. To date, there are also some 297 LAPACK Working Notes on the website.

LAPACK uses block-partitioned algorithms when possible and is structured on the BLAS, of course using Level 3 BLAS matrix–matrix operations whenever possible. Many people have been involved in LAPACK, as contributors, testers, and consultants. It has been a remarkable community project.[l] For more of the background, see Dongarra and Hammarling.[10,m] Jack had license plates "LINPACK" and "LAPACK" during the respective projects (see Figure 1), showing his devotion to the projects!

The BLAS and LAPACK are widely available in software packages and libraries so that software calling those routines can take advantage of efficient tuned versions. For example, NAG and MATLAB include them, as do chip manufacturers AMD, ARM, and Intel. LAPACK is at the heart of many application software packages, and so it must have millions of users, many of whom are probably not aware that they are using LAPACK.

FIGURE 1. LINPACK and LAPACK license plates.

The ScaLAPACK project was aimed at producing a version of LAPACK for distributed memory parallel machines. This required parallel versions of the BLAS, so the PBLAS were developed as part of the project, and a message passing library was also required. Jack and colleagues developed a library called PVM and, soon afterward, Jack was involved in a bigger community effort that produced MPI, now the main standard for message passing on parallel machines. ScaLAPACK had been designed so that it could readily use the different routines. ScaLAPACK was first released in 1995 and the manual was produced in 1997.[11] Again, royalties went to the SIAM Student Travel fund. Today, 25 years after its inception, ScaLAPACK is still a benchmark against, which new parallel codes are compared.

In early 1997 Clint Whaley, on his own initiative, started to work on autotuning and Jack soon recognized the importance of the work, which led to their seminal work on autotuning in the ATLAS project.[12] ATLAS enabled slow, laborious, hand tuning in the production and maintenance of high-performance compute kernels to be replaced by intelligent, self-adapting software, and this has transformed the way

[j]Also translated into Japanese.
[k]http://www.netlib.org/lapack/
[l]Sven regards it as the project he is most proud to have been involved in.
[m]Proceedings dedicated to James Hardy Wilkinson.

in which such kernel libraries are created and administered. The success of ATLAS in generating BLAS spurred manufacturers to improve the optimizations in their own offerings.

MIXED-PRECISION ALGORITHMS

Jack was quick to exploit the emergence in the 2000s of hardware in which single precision arithmetic was much faster than double precision arithmetic, notably Intel chips with SSE instructions and the Sony/Toshiba/IBM Cell processor, which gave speedups by factors about 2 and 14, respectively. Langou et al.[13] proposed to use single precision in computing the LU factors, but then using iterative refinement, which Jack of course was already familiar with, to compute the residual and update the solution in double precision in order to provide double precision results. They gave an error analysis to show that iterative refinement converges and is numerically stable as long as the system is not nearly singular to single precision. Since most of the work is in the LU factorization, which is carried out in single precision, this algorithm gave significant speedups, both in terms of computational speed and data movement. This work ushered in a new era of mixed-precision algorithms, which have proved to be particularly appropriate for GPU-based systems, with their mix of the half, single and double precision arithmetics.

OTHER PROJECTS

Jack has been involved with many other projects, a number of them at his own initiative. We mention just a few here. Netlib[n] went live in 1984 as a collection of public-domain mathematical software, papers, and databases, which has become a valuable resource to the scientific community. It was designed by Eric Grosse of Bell Labs and Jack at the suggestion of Gene Golub. Initially access was by electronic mail! At more or less the same time a newsletter digest and what became an online directory, the NA-Net project was started with the aim of fostering information and a sense of community among numerical analysts.[14] Both projects are still active today.

Since the turn of the century processor clock rates have stagnated and the cost of data movement has grown relative to the cost of floating point arithmetic. This necessitated a rethink in how linear algebra is performed on multicore processors and multisocket systems of multicore processors. Jack was at the forefront of the design and implementation of a new class of tile algorithms that break a computation into tasks and dynamically schedule the tasks on processors via an associated directed acyclic graph. These algorithms have been incorporated into the Parallel Linear Algebra Software for Multicore Architectures, 2008 (PLASMA)[o] linear algebra library. Originally built on the QUARK scheduler developed in Dongarra's group, PLASMA now uses OpenMP for scheduling.

Jack also produced the Matrix Algebra on GPU and Multicore Architectures (MAGMA)[p] library targeted at heterogeneous architectures, notably "multicore+GPU" systems. MAGMA splits algorithms into tasks of varying granularity and schedules their execution over the available hardware components. Small nonparallelizable tasks, often on the critical path, are scheduled on the CPU, and larger more parallelizable ones, often level 3 BLAS, are scheduled on the GPU. This library is the state of the art in the linear system and eigenvalue solvers for GPU systems.

Many modern applications are cast in terms of thousands of small matrix operations—a situation for which existing software, designed for large problems, is not efficient. Dongarra and coworkers have taken the lead in developing a Batched BLAS standard (2021) that describes a standard API that allows users to perform thousands of small BLAS operations in parallel, making efficient use of their hardware.[15] This article also proposes a standard for Batched LAPACK, such as the linear equation solvers. Batched BLAS have important applications in machine learning, for example.

Currently under development is Jack's Software for Linear Algebra Targeting Exascale, (SLATE, 2017–present)[q] library, which targets many-node HPC machines with large numbers of cores and multiple GPUs per node, and is intended to supersede ScaLAPACK. SLATE, as well as MAGMA, utilizes the Batched BLAS.

While program libraries have many advantages, scientists sometimes wish to develop their own codes—perhaps to solve problem variants, to specialize to particular computer architectures, or simply to experiment with and understand the underlying algorithms. Recognizing these demands, Dongarra and his co-authors published two collections (in both software and book form) of *templates:* pseudocode descriptions of general algorithms along with sample implementations, for

[n]http://www.netlib.org/

[o]http://www.icl.utk.edu/research/plasma
[p]http://www.icl.utk.edu/research/magma
[q]http://www.icl.utk.edu/research/slate

iterative methods for linear systems (1994) and, later in 2000, for eigenvalue problems.[r] These have been so successful that for many researchers they have become the *de facto* source for descriptions of these algorithms.

FURTHER COMMENTS

One remarkable facet of Jack's career has been his ability to maintain his Innovative Computing Laboratory (ICL) for well over 30 years. Most of the members have been supported by external grants, rather than being permanent members of the university. We are well aware that maintaining the pipeline of funding to support the group has been challenging, but the standard of work that ICL produces has always won through.

We also know that his biggest supporter is his wife, Sue, even though she sometimes has to remind him of who he really is at heart! Jack cherishes his family and, at the time of writing, Sue and Jack are the proud grandparents of eight grandchildren.

Jack has also been a big fan of the University of Tennessee Vols football team. On one visit, Sven was lucky enough to be taken by their son, Nick, to see a match, using Jack's season ticket as he was away. It was an amazing experience to be in a stadium of about 100,000 people watching a nonprofessional football match in Knoxville.

CONCLUDING REMARKS

Jack Dongarra's contributions to numerical linear algebra and HPC are pervasive. Every MATLAB user who solves a linear system by typing $A\backslash b$ is invoking the LAPACK linear equation solver. But, if they have the Parallel Computing Toolbox and they have the matrix A stored on a GPU, then they invoke the corresponding MAGMA code, also developed by Jack and colleagues. The same is true for many other mathematical software products, as well as numerous libraries used in today's software stack, because Dongarra's libraries are the state of the art for dense matrix computations.

Leadership comes not just from technical accomplishments like those mentioned previously, but also from helping the community to take stock of where it is and guiding it toward where it wants to be. Dongarra's TOP500 project has collected semiannual statistics on high-performance computer performance for almost 40 years and is used by the community to measure its progress over time and to identify trends in high-performance computation. He has introduced newer benchmarks, such as HPCG and HPL-AI to capture other aspects of performance, but TOP500 remains uniquely valuable. Jack regards it as his hobby.

As HPC has gone through the scales (megascale, gigascale, terascale, petascale, and now exascale), Jack has been there at every juncture, identifying what is needed to achieve the next level of performance and making the necessary algorithmic and software innovations. May he long continue to lead the way.

[r]https://www.netlib.org/linalg/html_templates/Templates.html and https://www.netlib.org/utk/people/JackDongarra/etemplates/book.html

REFERENCES

1. B. T. Smith, J. M. Boyle, B. S. Garbow, Y. Ikebe, V. C. Klema, and C. B. Moler, *Matrix Eigensystem Routines—EISPACK Guide*. Berlin, Germany: Springer, 1974, doi: 10.1007/3-540-07546-1.
2. J. J. Dongarra, J. R. Bunch, C. B. Moler, and G. W. Stewart, *LINPACK Users' Guide*. Philadelphia, PA, USA: SIAM, 1979, doi: 10.1137/1.9781611971811.
3. J. J. Dongarra and A. R. Hinds, "Unrolling loops in FORTRAN," *Softw.–Pract. Experience*, vol. 9, pp. 219–226, 1979, doi: 10.1002/spe.4380090307.
4. C. L. Lawson, R. J. Hanson, D. Kincaid, and F. T. Krogh, "Basic linear algebra subprograms for FORTRAN usage," *ACM Trans. Math. Softw.*, vol. 5, pp. 308–323, 1979, doi: 10.1145/355841.355847.
5. J. J. Dongarra, C. B. Moler, and J. H. Wilkinson, "Improving the accuracy of computed eigenvalues and eigenvectors," *SIAM J. Numer. Anal.*, vol. 20, pp. 23–45, 1983, doi: 10.1137/0720002.
6. J. J. Dongarra, "Improving the accuracy of computed singular values," *SIAM J. Sci. Statist. Comput.*, vol. 4, no. 4, pp. 712–719, 1983, doi: 10.1137/0904049.
7. J. J. Dongarra, J. Du Croz, S. Hammarling, and R. J. Hanson, "An extended set of FORTRAN basic linear algebra subprograms," *ACM Trans. Math. Softw.*, vol. 14, pp. 1–17, 1988, Algorithm 656, doi: 10.1145/42288.42291.
8. J. J. Dongarra, J. Du Croz, I. S. Duff, and S. Hammarling, "A set of level 3 basic linear algebra subprograms," *ACM Trans. Math. Softw.*, vol. 16, pp. 1–28, 1990, Algorithm 679, doi: 10.1145/42288.42291.
9. E. Anderson et al., *LAPACK Users' Guide*, 3rd ed. Philadelphia, PA, USA: SIAM, 1999. [Online]. Available: http://www.netlib.org/lapack/lug/, doi: 10.1137/1.9780898719604.

10. J. J. Dongarra and S. Hammarling, "Evolution of numerical software for dense linear algebra," in *Reliable Numerical Computation*, M. G. Cox and S. Hammarling, Eds., Oxford, U.K.: Oxford Univ. Press, 1990, pp. 297–327.
11. L. S. Blackford et al., *ScaLAPACK Users' Guide*. Philadelphia, PA, USA: SIAM, 1997, doi: 10.1137/1.9780898719642. (Includes a CD containing the software, an HTML version of the Guide and LAPACK Working Notes. An online version is at http://www.netlib.org/scalapack/slug/.)
12. R. C. Whaley, A. Petitet, and J. J. Dongarra, "Automated empirical optimizations of software and the ATLAS project," *Parallel Comput.*, vol. 27, no. 1/2, pp. 3–35, 2001, doi: 10.1016/S0167-8191(00)00087-9.
13. J. Langou, J. Langou, P. Luszczek, J. Kurzak, A. Buttari, and J. Dongarra, "Exploiting the performance of 32 bit floating point arithmetic in obtaining 64 bit accuracy (revisiting iterative refinement for linear systems)," in *Proc. ACM/IEEE Conf. Supercomputing*, 2006, p. 50, doi: 10.1109/SC.2006.30.
14. J. Dongarra, G. H. Golub, E. Grosse, C. Moler, and K. Moore, "Netlib and NA-Net: Building a scientific computing community," *IEEE Ann. Hist. Comput.*, vol. 30, no. 2, pp. 30–41, Apr. 2008, doi: 10.1109/MAHC.2008.29.
15. A. Abdelfattah et al., "A set of batched basic linear algebra subprograms and LAPACK routines," *ACM Trans. Math. Softw.*, vol. 47, no. 3, pp. 21:1–21:23, 2021, doi: 10.1145/3431921.

SVEN HAMMARLING is retired, but is a Senior Honorary Research Fellow in the Department of Mathematics, the University of Manchester, Manchester, M13 9PL, U.K. His long term interests have been numerical linear algebra and software implementing numerical algorithms. Contact him at sven.hammarling@manchester.ac.uk.

NICHOLAS J. HIGHAM is a Royal Society research professor and Richardson professor of applied mathematics in the Department of Mathematics, the University of Manchester, Manchester, M13 9PL, U.K. Much of his research is concerned with the accuracy and stability of numerical algorithms. His current research interests include mixed precision numerical linear algebra algorithms. Higham received his Ph.D. degree from the University of Manchester. Contact him at nick.higham@manchester.ac.uk.