Computer Science at Kent

The Collected Algorithms of the ACM

Tim Hopkins

Technical Report No. 4-08
Date: Dec 2008

Published by the Computing Laboratory,
University of Kent, Canterbury, Kent CT2 7NF, UK

The Collected Algorithms of the ACM

TIM HOPKINS
University of Kent, UK

December 2, 2008

Abstract

The Collected Algorithms of the ACM (CALGO) is now the longest running journal-published series of algorithms. After placing CALGO in the context of other journal algorithm series, we discuss the factors that we believe have made CALGO the well respected means of publishing mathematical software that it is today. We report on how moving with the times and technology has ensured the survival of CALGO, and we look briefly at how we may continue this in the near future.
Keywords: numerical software, software libraries, software quality, mathematical software.

1 Introduction

Collections of scientific routines have existed since the dawn of the digital computing age. Consisting of around 90 routines written in assembler, one of the first software libraries to be documented was developed around 1950 at the University of Cambridge for their EDSAC computer [56]. Authors of routines destined for this library had to demonstrate that their code was capable of solving a wide variety of problems and was efficiently programmed, well tested and properly documented. Even at this early stage the basic ground rules required for the successful take up of library software by end-users were recognized.

During the intervening years such collections have developed in many different directions: commercial general-purpose (NAG [43], IMSL [54]), institutional (CERN [10], Harwell [50]), application specific (*Applied Statistics* [47], *Computer Physics Communications* [17]), problem area specific (Eispack [51, 20], Lapack [2, 6]) and Open Source/Freely available (netlib [15], Gnu [21]). In all cases the driving force has been to provide end-users with reliable, accurate, efficient, general-purpose building blocks with which to construct or guide their own applications.

The Collected Algorithms of the ACM (CALGO) is unusual in that this collection was initiated quite early in the development of computer software and is still growing. While the implementation languages and the means of publishing a specific algorithm have both changed over the years, the basic ideals such as high quality, robustness and good documentation have remained and, indeed, improved with time and experience.

In section 2 we look briefly at the history of CALGO and discuss the collection in the context of a number of similar projects that have been undertaken by other journals. Section 3 looks briefly at how the choice of implementation languages has changed and how different journals influenced this choice. In Section 4, we review how the availability of the published software has changed and how this has affected the continuation of journal publication of software. Section 5 provides a discussion of the ways in which CALGO has influenced how

numerical software has evolved and how this evolution has influenced CALGO. Our conclusions are presented in Section 6.

2 History

Several journals have published software either as part of a series of algorithms or just as individual papers; these include *Applied Statistics* [47], *The Computer Journal* [8], *Numerische Mathematik* [53], *Numerical Algorithms* [52], and *Computer Physics Communications* [17]. In this section we consider how CALGO has changed over the last four decades and how other journals treated their series of published algorithms.

The CALGO series started in the February 1960 edition of *Communications of the ACM* (CACM) with the publication of a procedure for numerical integration [24]. As stated in the announcement for the new "additional department" [55], submitted procedures were to be implemented in Algol 60 [44] and submitted in the *publication form* of Algol. This latter requirement was to ensure that the printed form of the code was consistent and readable, for example, by the use of bold font to denote keywords instead of the variety of different quote marks that were used in practice. One major disadvantage of this requirement was that the code appearing in print had to be transliterated and rekeyed; an error-prone process. Although Algol procedures continued to be rekeyed, early Fortran codes appear to have been photo-reproduced from listings.

In 1975 with the advent of *ACM Transactions on Mathematical Software* (TOMS) and because the majority of algorithms being published in CALGO were numerical, CALGO moved to TOMS, where it has remained.

The two closest journal competitors to CALGO were *Applied Statistics*, whose algorithms appeared from 1968 to 1997, and *The Computer Journal*, where the algorithm series began in September 1964 and ceased in August 1987.

Like CALGO, *The Computer Journal* series also started very early; the first nine algorithms actually appeared in the *The Computer Bulletin* from September 1964 to December 1965 and the rest were published in *The Computer Journal*. This series can be viewed as a European competitor to CALGO; the application area, mainly numerical codes, was identical and the format of the published articles was also very similar to that used for CALGO. In total there were 123 published papers in this series; with the implementation languages showing a distinct bias towards 'European' languages, for example, Algol 68 and DAP Fortran!

Applied Statistics began its algorithm section in response to a report on a meeting on statistical programming [3] that had recognized the need for a specialist library because it felt that statistical software was not properly represented in other series, for example, CALGO and *The Computer Journal*.

Numerische Mathematik took a very different approach to publishing algorithm codes. While there was no single official series within the journal, algorithm papers (which included full code, a detailed description of the algorithm and the required parameters, a discussion of the numerical properties and sample test results) were prepublished ahead of their projected appearance in a series entitled *Handbooks for Automatic Computation*. Of the four handbooks envisaged for this series (Linear Algebra, Special Functions, Methods of Approximation and Integration) only the Linear Algebra volume appeared [57]. This volume contained 29 algorithms, 25 of which had appeared as fascicles.

The other three proposed books in the series failed to materialize presumably because of

the small number of published algorithms (special functions – 6, methods of approximation – 3, integration – 2 compared to 30 for linear algebra) and the rapid decline in the popularity of Algol 60 from the start of the 1970s. In addition to the Handbook series some 20 other algorithm codes were published in Numerische Mathematik between 1964 and 1971, all in Algol 60 and none explicitly designated as containing software.

3 Implementation Languages

CALGO mandated the use of Algol 60 [41] as its implementation language until the publication of the ASA standard for Fortran in 1966 [4], when the revised Algorithm Policy [25] allowed both languages. Surprisingly, the first published algorithm in Fortran did not appear until 1968 [58]. From 1970 onwards the use of Algol 60 steadily declined (the last Algol submission was in 1977 [16]) and the use of Fortran dominated. PL/1 was added to the list of approved languages in 1975 [18]. The approved list was replaced in future policies [34, 35, 36, 37] by a requirement for portability (see section 5 for a more detailed discussion). Any language is acceptable provided it has wide availability and the author adheres to the official language standard, if one exists, or is careful to use a subset of the language that is common to all implementations. Where they exist, the algorithm policy gives details of the acceptable standard for each language.

Adherence to the Fortran 77 standard was first required in the 1982 Algorithm Policy [36] although submissions checked by Pfort [49] (Fortran 66) were still acceptable. It was not until 1990 [37] that the Fortran 77 standard became the sole standard for Fortran. Fortran 2003 [32] is now the defining standard for Fortran although this contains almost all of Fortran 77 as a subset.

Apart from *Numerische Mathematik*, which only published software written in Algol 60, all other published series have allowed a variety of implementation languages and most require adherence to any available standards. Applied Statistics was probably the most prescriptive issuing an approved list of languages that changed with each new algorithm policy. Fortran 66 or 77 was the only language that was accepted throughout the series, with Algol 60 (until 1981), Algol 68 (1978-88), Cobol (1975-1979), Pascal and APL (1987-end) and PL/1 (1968-1975) also appearing. While emphasis was placed on standards conformance for the published software, this rule was relaxed for driver programs.

The question of whether to include languages on an approved list is always going to be fraught as trends come and go and few authors actually take advantage of non-mainstream languages. The CALGO solution of requiring the resultant code to be fit for purpose, widely portable and, if possible, standard-conforming is a good compromise.

Table 1 gives details of the languages that have been used to implement the algorithm series from *Applied Statistics*, *The Computer Journal* and CALGO. It also provides the number of algorithms accepted in each language and the date of the first appearance of the language in each series. This clearly shows the overwhelming use of Fortran and, in the early days, Algol 60 for the implementation of numerical software. It also illustrates how languages that have been greatly hypes to take over from Fortran have failed to gain a foothold in the numerical community, for example, just one algorithm has appeared implemented in Ada and none have yet appeared in Java. Given the Fortran community's continuing commitment to providing appropriate functionality for numerical computation, this seems unlikely to change in the near future.

Language	CALGO		Computer Journal		Applied Statistics	
	Number	Year	Number	Year	Number	Year
Algol 60	415	1960	81	1964	35	1968
Fortran (66,77)	379	1968	33	1968	274	1968
Pl/1	4	1973			1	1972
Non-standard Fortran	3	1977				
APL			4	1977		
Ratfor	1	1981				
Nitpack	1	1984				
Algol 68			1	1985		
Dap Fortran			1	1985		
Assembler	1	1987				
Pascal	1	1989	2	1984	9	1987
Matlab	25	1991				
C	16	1993			2	1993
Fortran (90+)	23	1994				
Lisp	1	1995				
C++	13	1997				
Ada	1	1999				
Awk	1	2000	1	1970		
Python	1	2004				
Total	886		123		321	

Table 1: Distribution of Algorithms by Implementation Language and First Year of Use

4 Availability of Algorithms

In the early days of journal algorithm series most of the published code was short; few of the first 250 algorithms that appeared in CALGO exceeded a few tens of executable statements. Users, therefore, felt no great hardship at having to rekey code; indeed there appears to have been almost a cottage industry in reporting the successful execution (or otherwise!) of many of the early CALGO routines. CACM published a large number of certificates and remarks on previously appearing algorithms, where users either confirmed the correct execution of the published code or reported the changes necessary to obtain working software.

However, as time progressed the submitted codes grew in size so that rekeying became less attractive, being both time-consuming and error-prone. In addition, publishers were aware that an increasing number of their published pages were being taken up with code listings.

ACM was almost alone in solving this problem; with the move to TOMS an algorithm distribution service was instituted that allowed potential users of published software to obtain machine-readable copies, initially, on magnetic tape and later on diskettes. At the same time, full code listings were discontinued from the journal and were made available in a separate CALGO supplement, initially as printed listing and then on microfiche before the supplement was discontinued completely in 2004.

Network access began in 1985 with the netlib service [15] allowing users to retrieve individual algorithms via electronic mail. All the available CALGO codes are now available via the ACM's Digital Library [1].

Providing on-line access to the algorithm submissions has meant that not only is code size

irrelevant but also that other material that might be useful to future implementors can also be included, for example, test drivers, data and expected results, makefiles and user manuals. Indeed the move to a free electronic repository for algorithm software and associated material was the key to the survival of CALGO. This was especially welcomed by authors, who now view publication in CALGO as a means of greatly expanding the user base of their software.

None of *The Computer Journal* codes was ever made available by the publisher. The codes from the *Applied Statistics* series were never officially made available in machine-readable form, although an incomplete set of routines has been obtainable via statlib [9] for many years. A collection of the 'best' routines was published as a book [22]; as with the journal, the sources, which had been reworked to provide a more consistent style, were presented in printed form.

On the other hand *Computer Physics Communications*, have provided access to their software first via magnetic tape and later on-line, although they have always levied a charge for access. Finally, *Numerical Algorithms* [52] accepts software as part of a submission; this is reviewed in parallel with the associated paper and accepted submissions are made available via netlib. Accepted software remains the responsibility of the authors, who are able to make corrections and upgrades to the published code.

5 The CALGO Experience

The algorithms that have appeared in CALGO cover a wide range of numerical topics. Table 2 provides a breakdown based on the IBM SHARE Library classification [29]; this illustrates well the diverse nature of the subject areas and also shows the popularity of linear algebra, special functions, integration and curve fitting and optimization. Statistical algorithms (G) account for almost 10% of the total although many of these (58/83) were published prior to 1975. Surprisingly, there has not been a surge of statistical algorithms since the demise of the *Applied Statistics* series in 1997.

While a published series of algorithms cannot hope to compete with a commercial library because it is not able to commission particular codes to generate the necessary depth of coverage of the field, it can, and does, provide an extremely useful service at other levels.

First, it helps to foster further research in particular areas, as the availability of state of the art codes means that other researchers have rapid access to the implementation details of the latest algorithms. Second, it allows application programmers to try out new methods and software on a wide variety of practical (and, therefore, typically difficult) problems. This can very quickly expose shortcomings in the new software or provide a springboard for wider acceptance of the new code. Indeed it is quite common for algorithms appearing in CALGO to find their way into commercial libraries. Third, they allow a useful sanity check on commercial algorithms.

One major disadvantage of publishing algorithms is that there is no simple way of unpublishing them if they are found to be defective or if they are superseded by better algorithms or implementations. A commercial library can just withdraw troublesome code by replacing it by a superior routine; while this may cause short-term problems to end-users in changing calls within their code, the benefits are obvious.

In many cases, published algorithms are downgraded because of improved understanding and more sophisticated analysis techniques. As a simple example consider the linear multiplicative random number generators that have appeared in CALGO; Algorithms 133 [7],

Numerical Area		Number of Routines
H	Operations Research, Graph Structure	56
D1	Quadrature	56
F4	Simultaneous Linear Equations	54
F1	Matrix Operations, including Inversion	51
Z	All Others	44
E2	Curve and Surface Fitting	42
F2	Eigenvalues and Eigenvectors of a Matrix	40
A1	Real Arithmetic, Number Theory	40
E4	Minimizing or Maximizing a Function	38
S14	Psi Function	32
G6	Permutations and Combinations	29
E1	Interpolation	27
D3	Partial Differential Equations	27
C5	Zeros of one or more Nonlinear Equations	27
S22	Miscellaneous Higher Mathematical Functions	26
D2	Ordinary Differential Equations	26
G5	Random Number Generators	25
M1	Sorting	20
C2	Zeros of Polynomials	18
S15	Polynomials, Hermite	16
J6	Plotting	15
G2	Correlation and Regression Analysis	15
C6	Summation of Series, Convergence Acceleration	15
S17	Bessel Functions of Real Argument	13
S20	Bessel and Related Functions, Miscellaneous	12
D5	Integral Equations	12
C1	Operations on Polynomials and Power Series	11
S13	Sine Integrals	10
G1	Simple Calculations on Statistical Data	10
S21	Theta Functions	8
S18	Modified Bessel Functions	6
R2	Symbol Manipulation	5
O2	Simulation of Computing Structure	5
F3	Determinants	5
E3	Smoothing	5
S16	Legendre Functions	4
G7	Subset Generators	4
F5	Orthogonalization	4
B4	Roots and Powers	4
A2	Complex Arithmetic	4
K2	Relocation	3
I5	Input – Composite	3
B3	Exponential and Logarithmic Functions	3
B1	Trig and Inverse Trig Functions	3
S19	Kelvin Functions	2
S03	Partitions	2
S	Approximation of Special Functions	2
D4	Differentiation	2
S23	Numerical Differentiation and Integration	1
M2	Data Conversion and Scaling	1
L2	Compiling	1

Table 2: Breakdown of CALGO algorithms by numerical subject area

266 [45] and 266R [23]. At the time of publication these were all considered to be useful additions to the collection. However, application of the spectral test [33, 27] produces the values given in Table 3. The requirement that $\{\mu_t\}_{t=2}^{6}$ should be greater than unity is obviously not attained by any of the generators and this indicates that none is suitable for general applications. They remain on the books as traps for the unwary, although the officially available code does contain health warnings in the comments!

Generator	a	m	Period	μ_2	μ_3	μ_4	μ_5	μ_6
CACM 133	5	2^{35}	$m/4$	$O(10^{-8})$	$O(10^{-7})$	$O(10^{-7})$	$O(10^{-6})$	$O(10^{-5})$
CACM 266	3125	2^{26}	$m/4$	1.83	0.68	0.53	1.53	2.11
CACM 266R	125	2^{26}	$m/4$	$O(10^{-3})$	0.49	2.36	0.82	5.80

Table 3: Spectral test results for CALGO random number generators

That publication in CALGO can provide the impetus for better algorithms is clear in a number of areas; for example, consider the computation of various special functions where numerous publications over the years have improved the overall accuracy and/or the range of values for which accurate results may be computed.

CALGO has always striven to publish high quality software. This is a moving target as the definition of quality, when applied to numerical software, has evolved over the years as our understanding of numerical applications and software engineering has improved. In addition, the term quality when applied to software may mean very different things to different people.

Prior to the move of CALGO to TOMS, algorithm codes were already being peer reviewed; authors were required to submit code either on cards or magnetic tape (tapes were returned!). The code was at least compiled and executed on some of the author provided test cases although it is not clear if this was performed by reviewers or the Algorithms Editor.

After 1975, reviewers received the code with the expectation that it was reviewed to the same standard as journal papers. Since the early 1990s the distribution of code to referees has routinely been via networks and the standard to which the software is reviewed has become increasingly rigorous. Authors are now required to submit extensive test programs which exercise all the features of their software along with files containing the expected run-time outputs. Referees are asked not only to check that the author supplied test programs execute correctly but also to perform additional testing of their own and to quality assure the code itself. In recognition of the effort involved in this process, the time allowed for review is longer that for normal research papers. As well as increasing exposure to the user community, authors recognize that acceptance of an algorithm into CALGO acts as a certification of their code and thus provides added status over publication via a personal web page.

In the beginning, authors publishing in CALGO were required to restrict the language facilities they used to those published in the then current Algol 60 reference manual [44, 41]. As more languages became available, portability became a requirement with strict adherence to available language standards. The onus was on authors to provide proof of the portability of their software; this could be by

- showing that code executed correctly on hardware using three different instruction sets, or
- using software tools to check for strict adherence to the language standard. (Authors of Fortran software could obtain copies of the pfort Fortran 66 standard conformance

checker [49] from the Algorithms Editor.)

CALGO thus attempted to further the use of software tools, something that many authors are still reticent to do today. The requirements for portability have no doubt ensured wide use of the software along with the longevity of the implementations.

Programming advice was, however, restricted to general software quality concerns, for example, the consistent naming of machine-dependent constants and adequate documentation of user-supplied parameters. Compared to other journals this advice was relatively low key; for example, contributors to the *Applied Statistics* algorithms series were faced with six pages of detail including advice on code layout, precision, label, jumps, and ifs, the construction of loops, etc. [48]. In retrospect CALGO was possibly too lax; for example, it never required user input to be checked, nor did it prescribe how problems detected during execution should be reported to the user, leading to major inconsistencies over the years.

The non-prescriptive approach to implementation languages has generally worked in that it has allowed authors both to experiment (not always successfully) with new languages (for example, Lisp and Python [39]) and to reflect popular use (for example, Matlab [26]). Implementation languages do, however, pose a number of problems.

First, languages fall into disuse, which can make future-proofing published codes extremely difficult if not impossible. Algol 60, for all its advantages as an implementation language over its nearest competitor Fortran, failed to gain the support of a number of major hardware manufacturers, most notably IBM, resulting in a rapid decline in its use. This meant that the wealth of algorithms developed in Algol during the 60s and early 70s quickly became unavailable unless they are translated into Fortran.

Fortran itself has evolved through a variety of standards [4, 5, 30, 31, 32]. However, much effort has been expended to ensure that, with each successive standard, software adhering to the previous standard is still, to a very large extent, valid. A project to modernize the Fortran codes that have appeared in ACM TOMS [28] aimed at using a small number of newer features to improve the portability of the software and to remove a large number of defects that could be detected using run-time checking facilities available with modern compilers. It did not attempt any major restructuring of the software.

While tools are available for automatically upgrading old Fortran code (nag_struct [42] and spag [46] both offer automatic improvements to the control structure and various tools are available for translating old fixed format sources into modern free format, for example, Spag [46]) it can be argued that using old code as a black box is likely to be cheaper and less error prone. This is what generally happens with library codes and, since only executables are usually provided, the user doesn't worry!

Testing and maintenance have always been important aspects in the development of CALGO. When software exchange was difficult, testing relied on individual users rekeying the code, developing their own test programs and data, and reporting their experiences via Remark and Certificate papers. Testing requirements have become far more formalized over the years and extensive testing material now forms a part of all algorithm submissions. This material should include not only simple example drivers to allow reviewers and users to check their implementation but also more stringent tests designed to exercise the code fully and to illustrate the range of problems that can be solved. All test material is included in the software bundle that is published in CALGO. We believe that this strict regimen has been one of the main reasons for the relatively low number of errors reported in published software.

The difficulty of generating fault-free software is well known and some formal mechanism

is necessary for users to report bug fixes and for authors to update their software. In the past this has been performed using formally reviewed Remark papers that, in the case of a bug fix, detail the source code changes required. All the criticism levelled at the publication of source code can be applied to Remarks – in the age of electronic publishing this is an outmoded mechanism. In the near future we expect to move CALGO into a source code control system like Subversion [11] and to use a system like Trac [40] to publish corrections and updates. All code updates will be summarised in TOMS in order to preserve the audit trail.

6 Conclusion

CALGO has seen the publication of algorithms that have had a major impact on the way in which we perform numerical computation; for example, the various families of Basic Linear Algebra Subroutines [38, 13, 12] and the use of a standard function for accessing machine dependent parameters [19].

Other series of published algorithms have met with similar success. The importance of the pre-publication of the Linear Algebra algorithms and the subsequent appearance of the Handbook for Linear Algebra cannot be underestimated. These codes set the standards that all writers of numerical software have since aspired to, and translated into Fortran, in some cases with minor improvements, they formed the core of the Eispack [51, 20] and Linpack [14] projects. In addition, both the Algol 60 version and the Fortran translations formed a major part of the first NAG subroutine libraries [43].

We believe that CALGO is still an important catalyst for furthering research in numerical computation as well as being an extremely valuable source of state of the art codes for use in applications. Authors of numerical software continue to recognize CALGO as a means of obtaining extensive peer review of their code as well as the underlying algorithm and, almost without exception, submitted software is improved by the review process. By making the sources of CALGO freely available electronically, the ACM has ensured that the published codes are accessible by a very wide audience; an added bonus for authors.

The move to electronic publishing is providing further opportunities to make the collection more dynamic and to allow for a more rapid and efficient mechanism for the reporting and fixing of software errors. We sincerely hope that, by continuing to move with the technology, CALGO will remain well respected and leading-edge in the decades to come.

References

[1] ACM. Digital Library, 2008. Available from: `http://www.acm.org/dl/` [cited 10 October 2008].

[2] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen. *LAPACK: Users' Guide*, volume 9 of *Software, Environments, and Tools*. SIAM, Philadelphia, third edition, 1999.

[3] Anonymous. Statistical programming: Prologue. *Applied Statistics*, 16(2):88–88, 1967.

[4] ANSI. *Programming Language Fortran X3.9-1966*. American National Standards Institute, New York, 1966.

[5] ANSI. *Programming Language Fortran X3.9-1978*. American National Standards Institute, New York, 1979.

[6] V. A. Barker, L. S. Blackford, J. Dongarra, J. Du Croz, S. Hammarling, M. Marinova, J. Waśniewski, and P. Yalamov. *LAPACK95: Users' Guide*. SIAM, Philadelphia, 2001.

[7] P. G. Behrenz. Algorithm 133: Random. *Commun. ACM*, 5(11):553, November 1962. http://dx.doi.org/http://doi.acm.org/10.1145/368996.368971 .

[8] British Computer Society. The Computer Journal, 2008. Available from: http://comjnl.oxfordjournals.org/current.dtl [cited 10 October 2008].

[9] Carnegie Mellon University. StatLib, 2008. Available from: http://lib.stat.cmu.edu/ [cited 10 October 2008].

[10] CERN – European Organization for Nuclear Research. CERN program library, 2008. Available from: http://cernlib.web.cern.ch/cernlib/ [cited 10 October 2008].

[11] Ben Collins-Sussman, Brian Fitzpatrick, and C. Pilato. *Version Control with Subversion*. O'Reilly Media, Inc., Sebastopol, CA, 2008.

[12] J. J. Dongarra, J. Du Croz, S. Hammarling, and I. S. Duff. Algorithm 679: A set of Level 3 Basic Linear Algebra Subprograms: model implementation and test programs. *ACM Trans. Math. Softw.*, 16(1):18–28, March 1990. http://dx.doi.org/http://doi.acm.org/10.1145/77626.77627 .

[13] J. J. Dongarra, J. Du Croz, S. Hammarling, and R. J. Hanson. Algorithm 656: An extended set of Basic Linear Algebra Subprograms: model implementation and test programs. *ACM Trans. Math. Softw.*, 14(1):18–32, March 1988.

[14] J. J. Dongarra, C. B. Moler, J. R. Bunch, and G. W. Stewart. *LINPACK: Users' Guide*. SIAM Publications, Philadelphia, 1979.

[15] Jack J. Dongarra and Eric Grosse. Distribution of mathematical software via electronic mail. *Commun. ACM*, 30(5):403–407, May 1987. http://dx.doi.org/http://doi.acm.org/10.1145/22899.22904 .

[16] T. M. R. Ellis and D. H. McLain. Algorithm 514: A new method of cubic curve fitting using local data [E2]. *ACM Trans. Math. Softw.*, 3(2):175–179, June 1977. http://dx.doi.org/10.1145/355732.355738 .

[17] Elsevier. Computer Physics Communications, 2008. Available from: http://www.cpc.cs.qub.ac.uk/cpc/ [cited 10 October 2008].

[18] L. D. Fosdick. Algorithms policy. *ACM Trans. Math. Softw.*, 1(1):5–6, March 1975. http://dx.doi.org/http://doi.acm.org/10.1145/355626.355629 .

[19] P. A. Fox, A. D. Hall, and N. L. Schryer. The PORT mathematical subroutine library. *ACM Trans. Math. Softw.*, 4(2):104–126, June 1978. http://dx.doi.org/http://doi.acm.org/10.1145/355780.355783 .

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

[21] GNU. GSL – GNU Scientific Library, 2008. Available from: `http://www.gnu.org/software/gsl/` [cited 10 October 2008].

[22] P. Griffiths and I. D. Hill, editors. *Applied Statistics Algorithms*. Ellis Horwood, 1985.

[23] L. Hansson. Remark on Algorithm 266. *Commun. ACM*, 9(9):687, September 1966. `http://dx.doi.org/http://doi.acm.org/10.1145/365813.365838` .

[24] R. J. Herbold. Quad I. *Commun. ACM*, 3(2):74, February 1960. `http://dx.doi.org/http://doi.acm.org/10.1145/366959.366964` .

[25] J. G. Herriot. Revised algorithms policy – August 1966. *Commun. ACM*, 9(9):683, September 1966. The doi associated with this reference is actually for Algorithm 290: linear equations, exact solutions, by J. Boothroyd. The revised policy is on the first page of this paper but has not been included as a separate document. `http://dx.doi.org/10.1145/365813.365822` .

[26] Desmond J. Higham and Nicholas J. Higham. *MATLAB Guide*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2000.

[27] T. R. Hopkins. AS193 – A revised algorithm for the spectral test. *J. Roy. Statist. Soc. Ser. C*, 32(3):328–335, 1983.

[28] T. R. Hopkins. Renovating the Collected Algorithms from ACM. *ACM Trans. Math. Softw.*, 28(1):59–74, March 2002. `http://dx.doi.org/http://doi.acm.org/10.1145/513001.513005` .

[29] IBM Users' Group. SHARE Reference Manual, 1973.

[30] ISO/IEC. *Information Technology – Programming Languages – Fortran (ISO/IEC 1539:1991(E))*. ISO/IEC Copyright Office, Geneva, 1991.

[31] ISO/IEC. *Information Technology – Programming Languages – Fortran - Part 1: Base Language (ISO/IEC 1539-1:1997)*. ISO/IEC Copyright Office, Geneva, 1997.

[32] ISO/IEC. *Information Technology – Programming Languages – Fortran - Part 1: Base Language (ISO/IEC 1539-1:2004)*. ISO/IEC Copyright Office, Geneva, 2004.

[33] D. E. Knuth. *The Art of Computer Programming Vol 2: Semi-numerical Algorithms*. Addison-Wesley, London, second edition, 1981.

[34] F. T. Krogh. Algorithms policy. *ACM Trans. Math. Softw.*, 4(2):97–99, June 1978. `http://dx.doi.org/http://doi.acm.org/10.1145/355780.355781` .

[35] F. T. Krogh. Algorithms policy (revised by W. Miller). *ACM Trans. Math. Softw.*, 5(2):129–131, June 1979. `http://dx.doi.org/http://doi.acm.org/10.1145/355826.355827` .

[36] F. T. Krogh. Algorithms policy (revised by R. J. Hanson). *ACM Trans. Math. Softw.*, 8(1):1–4, March 1982. http://dx.doi.org/http://doi.acm.org/10.1145/355984.355985 .

[37] F. T. Krogh. Algorithms policy (revised by R. Renka). *ACM Trans. Math. Softw.*, 16(3):293–296, September 1990. http://dx.doi.org/http://doi.acm.org/10.1145/79505.356315 .

[38] C. L. Lawson, R. J. Hanson, D. R. Kincaid, and F. T. Krogh. Algorithm 539: Basic Linear Algebraic Subprograms for Fortran usage. *ACM Trans. Math. Softw.*, 5(3):324–325, September 1979. http://dx.doi.org/http://doi.acm.org/10.1145/355841.355848 .

[39] Mark Lutz. *Programming Python: Object-Oriented Scripting*. O'Reilly & Associates, Inc., Sebastopol, CA, USA, 2001.

[40] D. J. Murphy. *Managing Software Development with Trac and Subversion*. Packt Publishing Ltd, Birmingham, UK, December 2007.

[41] Peter Naur, J. W. Backus, F. L. Bauer, J. Green, C. Katz, J. McCarthy, A. J. Perlis, H. Rutishauser, K. Samelson, B. Vauquois, J. H. Wegstein, A. van Wijngaarden, M. Woodger, and P. Nauer. Revised report on the algorithmic language ALGOL 60. *Commun. ACM*, 6(1):1–17, January 1963. http://dx.doi.org/10.1145/366193.366201 .

[42] Numerical Algorithms Group, Oxford, UK. *NAGWare f90 Tools (Unix)*, first edition, May 1995.

[43] Numerical Algorithms Group Ltd. NAG Fortran Library, 2008. Available from: http://www.nag.com/.

[44] Alan J. Perlis and K. Samelson. Preliminary report — International Algebraic Language. *Commun. ACM*, 1(12):8–22, December 1958. http://dx.doi.org/10.1145/377924.594925 .

[45] M. C. Pike and I. D. Hill. Pseudo-random numbers. *Commun. ACM*, 8(10):605–606, October 1965. http://dx.doi.org/http://doi.acm.org/10.1145/365628.365648 .

[46] Polyhedron Software, Oxford, UK. *plusFORT (Revision D)*, 1997. Available from: http://www.polyhedron.com/spag0html [cited 16 October 2008].

[47] Royal Statistical Society. Series C (Applied Statistics), 2008. Available from: http://www.blackwellpublishing.com/journal.asp?ref=0035-9254 [cited 10 October 2008].

[48] J. P. Royston, J. B. Webb, P. Griffiths, and I. D. Hill. Miscellanea: The construction and description of algorithms. *Applied Statistics*, 36(1):94–103, 1987.

[49] B. G. Ryder. The PFORT verifier. *Softw. Pract. Exper.*, 4(4):359–377, April 1974. http://dx.doi.org/10.1002/spe.4380150402 .

[50] Science & Technology Facilities Council. HSL, 2008. Available from: http://www.cse.scitech.ac.uk/nag/hsl/ [cited 10 October 2008].

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

[52] Springer Verlag. Numerical Algorithms, 2008. Available from: `http://www.springerlink.com/content/101751/` [cited 10 October 2008].

[53] Springer Verlag. Numerische Mathematik, 2008. Available from: `http://www.springerlink.com/content/100479/` [cited 10 October 2008].

[54] Visual Numerics Inc. IMSL Fortran numerical library, 2008. Available from: `http://www.vni.com/products/imsl/fortran/overview.php` [cited 10 October 2008].

[55] J. H. Wegstein. Algorithms: Announcement. *Commun. ACM*, 3(2):73, February 1960. `http://dx.doi.org/10.1145/366959.366964` .

[56] Maurice V. Wilkes, David J. Wheeler, and Stanley Gill. *The Preparation of Programs for an Electronic Digital Computer: Special Reference to the EDSAC and the Use of a Library of Subroutines*. Addison-Wesley, Reading, MA, USA, 1951.

[57] J. H. Wilkinson and C. Reinsch. *Handbook for Automatic Computation Volume II - Linear Algebra*. Springer-Verlag, New York, 1971.

[58] B. F. W. Witte. Jacobi polynomials. *Commun. ACM*, 11(6):436–437, June 1968. `http://dx.doi.org/http://doi.acm.org/10.1145/363347.363393` .