Is numerical software relevant? Is it too late to worry about quality?

James C. T. Pool
California Institute of Technology
Center for Advanced Computing Research, Pasadena, CA, USA.
Email: `jpool@cacr.caltech.edu`

Abstract
The long history and little progress of efforts to systematically evaluate and certify numerical software is recounted. Although this continues to be a serious problem, traditional numerical software libraries themselves are becoming increasingly irrelevant in today's computing environment. The reasons for this are described and some prescriptions for overcoming this crisis are presented.

Keywords
Archetypes, certification, evaluation, grand challenge applications, high performance computing, mathematical software, parallel software tools, templates

1 BACKGROUND

"Those who cannot remember the past are condemned to repeat it."
George Santayana, The Life of Reason, I/12 (1905-6)

The interest in quality of numerical software is a recurring event; indeed, the question of quality was a driving force in the emergence of the concept of mathematical software. For example, the NATS Project, which produced EISPACK and FUNPACK, commenced in 1971—25 years ago—two years after John Rice introduced the term mathematical software. The next year Wayne Cowell made the following introductory remarks at the Software Certification Workshop (Cowell, 1972), a workshop organized by Wayne Cowell and Lloyd Fosdick and attended by three participants in this Working Conference (Lloyd Fosdick, Leon Osterweil, and John Rice):

"Our concern at this workshop is with those processes by which [mathematical software] is produced, perfected, and made available to the user community. In addressing this problem, we will ask and seek answers to two kinds of questions. First are questions which are characterized as 'what' questions. What are the main issues and problems? What should be done about them? ... The second kind of questions are the 'how' questions. How can we organize ourselves so as to accomplish the work we see needs doing?"

Editions 1 and 2 of EISPACK had been released to the user community. Initial user experience had confirmed the validity of the NATS Project's production and testing processes. This success generated enthusiasm for planning a coordinated approach to numerical software. Workshop participants reported their experience in the NATS Project and numerous other projects. Based on these reports, they discussed the formation of an alliance of academic, government and commercial organizations dedicated to generating, evaluating, and distributing systematized collections of numerical software—with special emphasis on their evaluation. Although the alliance never emerged from the planning stage, many projects influenced by the discussions of the alliance produced numerical software, ranging from topical collections to comprehensive libraries, used widely in science and engineering applications. However, there was, unfortunately, a steady decline in the interest in the quantitative evaluation of numerical software. With few exceptions, these projects focused increasingly on developing and implementing new algorithms while treating evaluation as a necessary, but secondary, activity. Reasons for this decline include the difficulty of the problem, the lack of "glamour" of the problem, the need to provide software responsive to perceived user needs in a timely manner, and changing priorities in government basic and applied research programs. Bill Gear observed (Gear, 1979):

"Numerical software production is viewed by many people either as a routine programming task or as a by-product of that dull subject, numerical analysis, which itself falls somewhere between mathematics and computer science, too applied for the one and too irrelevant to the other."

In this "boring" field, documentation and evaluation are generally viewed by numerical software practitioners as its most boring aspects, but, since it is less visible, evaluation is easier to avoid!

2 THE CRISIS IN NUMERICAL SOFTWARE

Now, 26 years after the initial mathematical software conference at Purdue University organized by John Rice, the numerical software community faces a crisis. But it is not a crisis due to lack of quality! Rather the traditional delivery mechanism, the library (or topical collection) of quality software, is increasingly irrelevant in today's computing environment. The library's role has been eroded by

- publications such as the Numerical Recipes series,
- general computational systems such as MATLAB,
- application specific computational systems such as NASTRAN,
- the application developer's return to a "do it yourself" approach for parallel computing systems, and

- pricing in the distributed environment including low cost workstations and personal computers.

One need only observe an application oriented Internet group, for example, the Computational Chemistry List (CCL), or browse the bookshelves of scientific and engineering colleagues to appreciate the impact of Numerical Recipes. The interactions between numerical software developers and computational system developers have been minimal for non-technical reasons.

Most developers of computational systems have reimplemented algorithms into their systems to avoid intellectual property and liability issues. The report of Shampine and Reichelt on the development of ODEs for MATLAB is an encouraging example (Shampine, 1996) as is the availability of the NAGware Gateway Generator for interfacing NAG Library routines to MATLAB and the MAGNum collection of NAG Library routines provided with MATLAB interfaces.

A recent informal survey of several Grand Challenge Application Groups (GCAGs) of the US High Performance Computing and Communication (HPCC) Program revealed GCAGs utilize little, if any, numerical software developed outside their group. The numerical software developer and the developer of applications for parallel computer systems must interact more—blackbox software is not adequate to solve problems on the frontier of science and engineering. Almost 20 years ago, Reid and Hopper observed (Reid, 1979):

> "It may be argued that if a piece of software is well tested and well documented and in use as a 'black-box' in compiled form, then there is no need for the source to be readable. Indeed it can even be argued that it is better if it is unreadable for then the user is likely to be frightened away from the temptation of making any changes.
>
> Our view, however, is that the advantages of clear well-structured and well-commented source are overwhelming. To expect the documentation to specify the finest details of the algorithm is not realistic; if this is wanted there is really no substitute for looking at the source code itself. Indeed the source may be regarded as an important part of the documentation."

The introduction of templates takes this view to another level. Barrett, et al. introduced their book on templates for iterative linear system solvers with the following question (Barrett, 1994):

> "Which of the following statements is true?
>
> - Users want 'black box' software that they can use with complete confidence for general problem classes without having to understand the fine algorithmic details.
> - Users want to be able to tune data structures for a particular application, even if the software is not as reliable as that provided for general methods.
>
> It turns out both are true, for different groups of users.
>
> Traditionally, users have asked for and been provided with black box software in the form of mathematical libraries such as LAPACK, LINPACK, NAG, and IMSL. More recently, the high- performance community has discovered that they must write custom software for their problem. Their reasons include inadequate functionality of existing software libraries, data structures that are not natural or convenient for a particular problem, and overly general software that sacrifices too much performance when applied to a special case of interest.

Can we meet the needs of both groups of users? We believe we can. Accordingly, in this book, we introduce the use of templates. A template is a description of a general algorithm rather than the executable object code or the source code more commonly found in a conventional software library."

While few, if any, GCAGs use templates developed by the numerical software community, the concept of templates received enthusiastic support of "grand challenge users" at the First Pasadena Workshop on System Software and Tools for High-Performance Computing Environments (Fox, 1992). In practice, the GCAGs start from algorithmic descriptions, or often source for serial programs, and proceed to use this information quite like the proposed template-based process. However, because the GCAGs are focused on scientific or engineering results rather than providing software, their approach differs. This is not a new phenomenon—almost 20 years ago, Bill Gear commented (Gear, 1979):

"By and large, most big numerical codes are not written by numerical analysts, or even computer scientists, but by engineers, physicists, and other large computer users. These people tend to underestimate the difficulty of producing reliable code, but in spite of this, or perhaps because of it, they have been responsible for most important methods that have been developed in the past .... When these people had a real problem to solve, they could not afford to be deterred by minor mathematical difficulties, so they invented new methods. There is a tendency for numerical analysts and computer scientists to ignore or disparage the accomplishments of the writers of large problem-oriented packages .... "

Unfortunately, the US HPCC agency program managers who defined the GCAG Program recognized the above and urged proposers to have multidisciplinary research teams. Recent reviews of the GCAGs have confirmed anticipated benefits of the multidisciplinary composition of the research teams, typically a combination of application scientists or engineers together with computer scientists and applied mathematicians. Continuing Gear's comment:

"Many of the large codes utilize a fine blend of 'engineering insight' and applicable theory to obtain results that could not be obtained by a numerical analyst or computer scientist. ... Part of the challenge of numerical software is to produce codes which can be embedded within large packages to handle standard operations such as the solution of differential equations. Although some modern numerical software is far more reliable than the corresponding sections of the large problem-oriented packages, these packages do not generally use library software because the latter is insufficiently flexible to be tailored to a particular class of applications and still retain efficiency."

The GCAGs were an opportunity for the numerical software community to collaborate with scientists and engineers and explore new approaches to providing numerical software for large-scale computational science and engineering; however, there is a conspicuous absence of the numerical software community from almost all GCAGs. In contrast, the recently initiated PINEAPL Project (Parallel Industrial Numerical Applications and Portable Libraries Project, coordinated by the Numerical Algorithms Group Ltd. under the Fourth Framework Programme of the European Commission's Information Technologies RTD Programme) involves both end users with major application programs and numerical software developers. The application programs include electromagnetic, fluid dynamics, chemical reaction, and thermal problems, while the numerical software includes dense and sparse linear algebra, ordinary and partial differential equations, and optimization.

3 NEW APPROACHES

Despite the lack of participation of the numerical software community in the GCAGs, can we learn for the GCAGs' experience? It is intriguing that several GCAGs developed a problem-solving environment to facilitate their efforts. This raises the question whether PSEs can become an important delivery mechanism for numerical software and, in particular, provide a framework for tailoring templates or archetypes in parallel computing environments. Answering the latter question is a primary objective of a recently initiated NSF Multidisciplinary Challenge, the PSEware Project, a collaboration among applied mathematicians, computer scientists, and application scientists and engineers at the California Institute of Technology, Drexel University, Indiana University, Los Alamos National Laboratory, New Mexico State University, and the University of California at Irvine (see `http://www.extreme.indiana.edu/pseware`). for a brief description of the PSEware Project). Archetypes developed at Caltech will be used with object-oriented numerical tools developed at LANL and NMSU in conjunction with PSEware components to solve, for example, PDE problems posed by LANL scientists.

If the numerical software community can find ways to work with developers of computational systems, both general systems and application specific systems, and with application developers, then the evaluation of numerical software will again become a critical issue. Knowledgeable developers of computational systems will demand better understanding of the domain of applicability of numerical software used in their system. (An increasingly important challenge to the numerical software community is the numerically naive developer of mass market products, e.g., plug-ins for spreadsheets.) Assurance of quality becomes imperative when these computational systems are used by scientists and engineers, quite expert in their respective fields, but naive about numerical computation. Likewise, a new approach to evaluation will be required if alternative delivery mechanisms such as templates, rather than executable software, are adopted to satisfy the "grand challenge user."

There are trends in high-performance computing which suggest questions of numerical software quality will be simplified. For example, there is widespread belief that the emergence of shared memory systems will simplify parallel programming. The euphoria over shared memory parallel systems will undoubtedly fade as these systems are clustered to form larger and larger systems to solve large-scale problems on the frontier of science and engineering. The complexities of memory hierarchy and the variety of associated transfer rates will make algorithm development and tuning extremely subtle tasks. Thus, one can anticipate new challenges in the evaluation of the quality of numerical software instead of simplification.

This Working Conference on the Quality of Numerical Software: Assessment and Enhancement provides an opportunity to revitalize interest in the evaluation of numerical software. It coincides with a growing recognition that both the assessment and enhancement of system software and tools, including numerical software, for high performance computing environments is critical for the future evolution of the US HPCC Program; indeed, the primary topic of the upcoming Workshop on Software Tools for HPC Systems is the role and organization of a proposed system software and tools testing and evaluation center. It also coincides with more empirical approaches, in particular, web-based user-provided evaluation derived from usage of system software and tools, planned by both the National HPCC Software Exchange, a project of the Center for Research in Parallel Computation sponsored by US HPCC agencies (see `http://www.netlib.org/nhse/`) and the Ptools Consortium (see `http://www.ptools.org`).

However, the similarities between the reports of experience at the Software Certification Workshop in 1972 and those presented at this workshop are disconcerting. While there have been evolutionary advances on various technical fronts in evaluating the quality of numerical software, there does not appear to have been any fundamental technical achievements or successes that permit progress in new directions. Acknowledging that evaluation of the quality of numerical software is an extremely difficult technical problem, one would at least expect progress on organizational and other issues. Again quoting Wayne Cowell (Cowell, 1972),

> "I believe that there was a consensus that ultimately there is a need for an independent certifying agency, that is, independent of the producers of the software. Both the evaluation of a routine and the validation of its documentation should be done by this independent agency. It was admitted that this situation does not exist yet today. Indeed people like the NATS group are, in a sense, certifying their own work . . . ."

Reports in this workshop from the descendants of NATS, including LAPACK and ScaLAPACK, demonstrate this situation has not changed. The need for multidisciplinary interactions was also recognized 25 years ago (Cowell, 1972):

> "There was a brief discussion as to what the best interface between users and experts was. The most widely acclaimed interface was through small specialized group meetings which focused on problem areas rather than around professional alignments."

However, now the numerical software must adopt a multidisciplinary approach and actively participate in the development of applications to find new modes for delivering numerical software to the end user; otherwise, numerical software will become increasingly irrelevant and the question of quality of numerical software will be extraneous.

REFERENCES

Barrett, R., et al. (1994) *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods*, SIAM, Philadelphia.

Cowell, W., ed. (1973) *Proceedings of the Software Certification Workshop*, Snow Mountain Ranch, Granby, Colorado, August 27-30, 1972, Technical Report, Argonne National Laboratory.

Fox, G., et al. (1992) Impact of Grand Challenge Applications on HPCC Software and Tools, in *System Software and Tools for High-Performance Computing Environments*, (eds. P. Messina T. and Sterling), SIAM, Philadelphia, 25–33.

Gear, C. W. (1979) Numerical Software: Science or Alchemy?, Technical Memorandum UIUCDCS-R-79-696, Dept. of Computer Science, University of Illinois at Urbana/Champaign.

Reid, J. K., and Hopper, M. J. (1979) Production, Testing and Documentation of Mathematical Software, *CSS 75*, Computer Science and Systems Division, A.E.R.E. Harwell.

Shampine, L. F., and Reichelt, M. W. (1997) Developing ODE Software in New Computing Environments, in *The Quality of Numerical Software: Assessment and Enhancement*, (ed. R. F. Boisvert), Chapman & Hall, London. This volume.

DISCUSSION

Speaker : J. C. T. Pool

J. Rice : *Grand* challenges sell well in getting new government research programs started and funded. However, I believe the major impacts are made by being able to solve moderate sized problems, call them *petty* challenges, easily, reliably, even automatically. One can package numerical software to solve such problems. It is unrealistic to hope that pre-packaged software can solve a significant part of a grand challenge problem; this will happen very rarely. The most pressing need is to find mechanisms to get high quality numerical software into the mass markets. Forty years ago everyone was writing their own code for the logarithm. That happens rarely now. Such computations are incredibly more reliable, and some real progress has been made.

J. C. T. Pool : While I concur that the traditional approach to numerical software will not help the Grand Challenge Application Groups, the Grand Challenge Application Groups potentially provide the numerical software community with a testbed to explore new approaches to providing numerical software, especially for future generations of parallel computing systems. We have repeatedly learned that our experience in solving the most difficult problems has a "trickle down" effect to the small-to-moderate sized problems.

The authors of Numerical Recipes have demonstrated a way to introduce numerical software to a larger user communities. Whether a variation on their approach is viable for commercially supported high quality numerical software has not been answered. Neither the Numerical Algorithms Group nor Visual Numerics, the primary independent software vendors (ISVs) providing high quality numerical software, have achieved their level of market penetration using the traditional library approach. On the other hand, there is no question that increasing numbers of users are depending on high quality numerical software from ISVs and the public domain.

S. Wright : I believe that the Internet and web present us with a new and important delivery mechanism for numerical software. The NEOS project at Argonne National Laboratory (ANL), in which we deliver information and remote solver tools to users of optimization software via the Internet/web, is being heavily accessed, according to the "number of hits" metric. We believe that the NEOS model of "information server" and "compute server" is also appropriate for other areas of numerical software, at least for users at the "retail" level who wish to solve small-to-medium sized problems. See the NEOS web site at `http://www.mcs.anl.gov/home/otc/`.

J. C. T. Pool : ANL's NEOS Project and the Center for Research in Parallel Computation 's National HPCC Software Exchange are two very interesting examples of experiments to use the Internet and web technology to disseminate information about software. The experiments with "compute servers" are especially interesting, but open a variety of licensing issues for commercially supported software in addition to technical interoperability issues.

W. V. Snyder : My experience with "real" users makes me skeptical of your assertion that users of computational software by way of integrated problem solving environments will demand quality. They seem, by and large, to be happy to get the correct answers "most of the time."

J. C. T. Pool : I never promised you a rose garden! Interactions with "real users" are, indeed, thorny. However, the numerical software community needs the interaction with these users if it is to remain viable.