MINRES-QLP Pack and Reliable Reproducible Research via Staunch Scientific Software*

Sou-Cheng T. Choi†

choi-wssspe-10.tex, v10, revised Sat Sept 07 10:44:27 UTC+8

Abstract

MINRES-QLP Pack is a suite of (extended) Krylov subspace methods for solving large linear systems and linear least-squares problems possibly with singular or ill-conditioned matrices or linear operators with special symmetries. Our purpose is to create and develop robust open-source implementations of the associated algorithms in MATLAB, Fortran, and PETSc that are faithful to the theory, following the philosophy of reproducible research and practicing development principles of staunch scientific software (SSS) that promote reliable reproducible research (RRR).

In this paper, we review the key features in the ongoing theoretical and software development of our algorithms in MINRES-QLP Pack. We highlight the most effective software engineering tools known to us, that are potentially useful to other scientific research areas. We contend that the principles of RRR and SSS should be propagated to computational science students in advanced courses of scientific computing. To this end, we have started an experimental seminar course, "Reliable Mathematical Software" (IIT MATH-573) in our institution. We will share our research practice and pedagogic experiences in the Workshop on Sustainable Software for Science.

1 Introduction

Reproducible research (RR) in computational sciences was pioneered by Claerbout in the Stanford Exploration Project (SEP) [13]. It has since been championed by Claerbout, Donoho, and their collaborators in their research areas of geophysics, signal and image processing. The leading advocates for RR include Gentleman and Peng in biostatistics, Koenkera in economics, LeVeque in numerical partial differential equations, and Stodden in legal frameworks that enable RR. The defining principle of RR [7] is "When we publish articles containing figures which were generated by computer, we also publish the complete software environment which generates the figures." Clearly the dependency of RR to software, hardware, code, and data are inevitable.

We have two main goals in this paper. First, we contend that the bases of worthy RR in modern computational sciences have to be strong mathematical, statistical, and approximation theories leading to rigorous error and cost analysis. Second, we establish our position that RR can be made substantially more reliable—hence reliable reproducible research (RRR)—by staunch scientific software (SSS), which we define as *a conceptual framework that encompasses a collection of software development methods for promoting the reliability of reproducing provably correct results in computational sciences.*

While RRR and SSS are not entirely new, they are not as often present as hoped. Relatively few computational scientists seem to employ them or even practice RR as reflected in recent survey results put together by Flemisch [15]. When SSS is absent, RR risks remaining an ideal. On the other hand, conscientious adoption of the principles can lead to substantially more trustworthy research results.

More specifically, in terms of theory, we have a relatively new suite of algorithms for solving singular or ill-conditioned linear systems of equations or linear least-squares problems with guarantee of accuracy. In terms of practice, we do what we advocate; we borrow a number of strategies and tools from industrial software engineering that lead to more reliable RR.

The outline of this paper is as follows. The next section gives a compressed description of MINRES-QLP Pack's algorithms. Section 3 recapitulates principles and benefits of RR, including examples of high-quality software and associated highly cited publications. In Section 4, we draw from our computational research experiences and highlight a number of strategies and tools for robust software package development. These include our initiative of conducting a weekly seminar course, which serves to educate research students about RRR and SSS. In the last section, we conclude with thoughts for future work.

*This work was supported in part by grants from the National Science Foundation under grant SES-0951576, the University of Chicago Energy Initiative.

†Computation Institute (CI), University of Chicago and Argonne National Laboratory (sctchoi@ci.uchicago.edu); Department of Applied Mathematics, Illinois Institute of Technology (schoi32@iit.edu).

2 MINRES-QLP Pack

Most Krylov methods for solving large linear systems or linear least-squares problems usually assume nonsingular or full-rank matrices. They are generally divided into two classes: those for symmetric matrices (e.g., conjugate gradient, MINRES, SYMMLQ), and those for unsymmetric matrices (e.g., BCG, GMRES, QMR, BICGSTAB, LSQR, and IDR(s)). Such a division is largely due to the fact that historically the most prevalent matrix structure from practical applications is symmetric. However, other types of symmetry structures, notably complex symmetric, skew symmetric, and skew Hermitian matrices, are becoming increasingly common in modern applications. Often, these are treated as general unsymmetric matrices.

On a singular linear system such as $\left[\begin{smallmatrix}1 & 0\\ 0 & \varepsilon\end{smallmatrix}\right] x = \left[\begin{smallmatrix}1\\ 1\end{smallmatrix}\right]$, where ε is the machine precision, the pseduoinverse solution should be $x^\dagger = \left[\begin{smallmatrix}1\\ 0\end{smallmatrix}\right]$. Most of MATLAB's Kyrlov solvers abort or return an exploding solution. To carefully handle singularity and exploit various symmetry structures, we have designed a suite of MINRES-QLP [11, 9] algorithms, which can constructively reveal (numerical) singularity and compatibility of a given linear system of equations; users do not have to know these properties a priori.

The key automatic steps of a MINRES-QLP algorithm are: **first**, detect special symmetry with an economical but effective statistical test. **Second (optional)**, construct a symmetric positive definite preconditioner M whose number of distinct nonzero eigenvalues $M^{-1/2}AM^{-1/2}$ is less than that of A and hence number of iterations required in the next stage will be reduced. Without loss of generality, let M be identity in our discussion. **Third**, in the kth iteration, for $k = 1, \ldots, \min\{p, K\}$, where p is the smallest positive integer such that $\{b, Ab, \ldots, A^p b\}$ is linearly dependent, and K is a user-input positive integer, the algorithm searches for x_k in the kth Kyrlov subspace $\mathcal{K}_k(A, b) =< b, Ab, \ldots, A^{k-1}b >$, where x_k is the minimum-length element of minimal-residual solutions for the problem $\min_{x\in\mathcal{K}_k(A,b)} \|Ax - b\|$. It is known that x_k exists and is unique. If either residual norms $\|r_k\| = \|b - Ax_k\|$ or $\|Ar_k\|$ is smaller than a scalar multiple of a user-given tolerance ϵ, then the algorithm returns x_k as an approximant; otherwise the third step is repeated with k incremented by one. We note that the square root $M^{-1/2}$ is not explicitly constructed and all the quantities such x_k, $\|r_k\|$,$\|Ar_k\|$ are accurately and efficiently computed by short recurrences.

In MINRES-QLP Pack, our goals are to have MATLAB routines well-documented, well-tested, optimized for speed, accompanied by examples, and stored in a repository [5] where they can be modified, extended, and re-tested as needed. Over time we will work to improve existing algorithms and add new methods in the package. If resources permit, we will implement them in other languages such as Fortran [12] and PETSc [2].

3 Reproducible research

RR by Claerbout is not simply a form of ideological rhetoric. It actually has specific instructions. Claerbout encouraged all authors of computational research papers to mark each figure result with the tags **[ER]** for "easily reproducible", **[CR]** "conditionally reproducible", or **[NR]** "non-reproducible" to indicate the extent of reproducibility. To be qualified for **[ER]**, a figure is generated and accompanied by programs with parameters and input data, as well as "makefiles executable on Linux and portable to other operating systems." A conditionally reproducible figure are often time consuming to reproduce from scratch and depends on the availability of computing resources such as memory, commercial applications, or large input datasets. Non-reproducible figures may include images drawn or photographs taken—not computed—to illustrate scientific ideas. Today there is no reason not to extend the tags to other computational results or online resources such as interactive figures, video clips, tables, or online databases produced or referenced by a report [14].

Paraphrasing Claerbout, Donoho stated in Buckheit & Donoho 1995 that "an article about computational result is *advertising*, not scholarship. The actual **scholarship** comprises the **full software environment**, **code** and **data**, that produced the result." Donoho et al. produced multiple MATLAB packages, WaveLab (1995), BeamLab (2004), SparseLab (2007), and MCALab (2008). They are associated with some of the highly cited papers, for example, [8, 17]. It is not an unreasonable hypothesis that quality reproducible software contributes significantly or correlates positively to the citation statistics of the associated reports.

Nevertheless, we ask in practice, how reliable is RR? It would be at most as much as the strength of its underlying theory, data, code, software, and hardware. Obviously, it is a "decreasing function" of time due to emergence of new software (versions), phasing out of old computer architecture, affordability of commercial software, or availability of quality free software. Often, authors release only partial data, code fragments, or incomplete environment specification. It is not uncommon to see loss of data or code due to insufficiently frequent backup or unstable transfer processes. Algorithms of suboptimal design, with insufficient documentation,

inadequate testing, and infrequent release of bug fixes—all understandable due to limited human resources or expertise—may render the computational software practically unusable even to the original creators not too much later in time. Today's complex applications with big data and big code only add to these difficulties.

4 Staunch scientific software

Flemisch's survey [15] on reproducible research received feedback from only about 10% of targeted recipients. Among those who responded, more than 70% reported that the efforts to reproduce others' or even own computational results are more than little. It also pointed out that RR is hard work but perceived as "unrewarding". The survey suggested the roles of strategic tools such as the use of a repository (e.g., SVN, CVS, GIT). The survey results are in line with our understanding and experiences. Despite the burden of RR, we maintain our position and belief in the importance of establishing reliable RR via staunch scientific software. We consider three areas to address the challenges: tools, incentives, and education.

Tools. From our practices of reproducible research in our work, we affirm that *RRR via SSS can be done.* Many small specific examples already exist even though it is probably extremely hard to find "a general solution" for a given scientific field. The tools we highlight are chosen for their compatibility with MATLAB and effectiveness of assisting researchers to developing correct and polished algorithms. In addition to the use of a code repository, we also employ Integrated Development Environment (IDE) software. In the case of MATLAB, it has its own IDE built in Java, with visually appealing automated reports on code dependency, documentation evaluation, and performance analysis, which together help pinpoint at line level where the software can be improved. We emphasize on the importance of designing simple application programming interfaces (APIs) from the outset. If we were allowed to communicate only one thing to expert or application users about our implementation, our choice would be its API, which consists of function name, required and optional inputs, outputs, and our parsing schemes. Our algorithms parse user inputs and ensure that they fall in the correct ranges; in some cases we supply default values for missing optional parameters and gently correct invalid input values. We craft detailed and readable documentation in both text and searchable HTML formats with specification of APIs, syntax, default input values, examples of usage, and further references. Last but not least, to ensure the correctness of computational results from our software and examples in the documentation, we create unit tests [4] and doctests [3]. An effective unit test or doctest should be demonstrative of usage, and run in less than a small fraction of a second. Without these tests in place, our experience is that even minor changes to our code or documentation could quickly and unknowingly depart from each other and theory. These two kinds of tests effectively safeguard most changes we make to our algorithms. *Every time before code is checked into the repository, unit tests and doctests are run. Every time a bug is found, unit tests are expanded.* All unit tests and doctests have to pass at 100% success rate for each code check-in and software release.

Incentives. Practitioners of RRR and SSS need more incentives in terms of specialized software citation [16], comprehensive journal review policies for software publication such as ACM TOMS' [1], and research funding schemes such as NSF's SSE & SSI program[6] that gives significant weightage to quality scientific software infrastructure at all scales. To maximize quality software's exposure and usage, we support free software and open source movement, but are also open to having our software included in quality commercial software.

Education. This Summer, we started a team devoted to development of a MATLAB research software package GAIL [10] for numerical integration with guarantee of accuracy. Our members role play a team of software engineers with specializations in three of the critical software functions, namely testing, documentation, and release. We make use of a wiki to strengthen team communication, recording key steps of important scientific routine process and procedures; maintaining checklists of day-to-day tasks; and sharing of intermediate results, knowledge, and expertise. We prepare our students of applied mathematics to become practitioners of RRR via SSS. As a beneficial side effect, as they have gained software engineering skills that are usually obtainable only in information-technology companies, they become more attractive job candidates. In fact, one of our five student team members has been offered a part-time position as a software testing engineer.

We say, "*Those who can, do and teach.*" Principles and novel ideas behind the research work should ideally be propagated in beginning and advanced courses in scientific computing, whereas the practices of engineering staunch software promoted to computational scientists and researchers through workshops such as WSSSPE [18]. To this end, we have started an experimental seminar course, "Reliable Mathematical Software" (IIT MATH-573) in our home institution. We will draw teaching materials from exemplary research papers accompanied with quality software.

5 Conclusions

We agree with the importance of reproducible research at all scales, and recommend to strengthen it with staunch scientific software development practices. In the short term, RRR via SSS may seem more time consuming. In the long run, the investment of time and efforts gives the software a better chance to "survive" and make a greater impact in the research area. Some researchers may actually find that our approach of RRR via SSS provides to some extent a structured course for investigating and attacking complex computational problems. More reliable incremental research can be built on staunch software packages, leading to stably upward spirals of progress in scientific knowledge. As a community, we need more time to reflect, brainstorm, and experiment specific operations. We also need more ongoing dialogue on the topic through courses, seminars, workshops, and conferences.

References

[1] ACM Algorithms Policy. http://toms.acm.org/AlgPolicy.html.

[2] PETSc: Portable, Extensible Toolkit for Scientific Computation. http://www.mcs.anl.gov/petsc/.

[3] Doctest. http://www.mathworks.com/matlabcentral/, 2010.

[4] MATLAB Unit Testing Framework. http://www.mathworks.com/help/, 2013.

[5] MINRES-QLP Project and Community Website. http://code.google.com/p/minres-qlp/, 2013.

[6] NSF 13-525: Software Infrastructure for Sustained Innovation - SSE and SSI. http://www.nsf.gov/pubs/2013/nsf13525/nsf13525.htm, 2013.

[7] J. B. Buckheit and D. L. Donoho. *Wavelab and reproducible research.* Springer, 1995.

[8] S. S. Chen, D. L. Donoho, and M. A. Saunders. Atomic decomposition by basis pursuit. *SIAM Journal on Scientific Computing*, 20(1):33–61, 1998.

[9] S.-C. T. Choi. Minimal residual methods for complex symmetric, skew symmetric, and skew Hermitian systems. Tech. Rep. ANL/MCS-P3028-0812, CI, University of Chicago, Chicago, IL, 2013.

[10] S.-C. T. Choi, Y. Ding, F. J. Hickernell, L. Jiang, and Y. Zhang. GAIL: Guaranteed Automatic Integration Library (Version 1), MATLAB Software. http://code.google.com/p/gail/, 2013.

[11] S.-C. T. Choi, C. C. Paige, and M. A. Saunders. MINRES-QLP: A Krylov subspace method for indefinite or singular symmetric systems. *SIAM J. Sci. Comput.*, 33(4):1810–1836, 2011.

[12] S.-C. T. Choi and M. A. Saunders. ALGORITHM: MINRES-QLP for singular symmetric and Hermitian linear equations and least-squares problems. *ACM Trans. Math. Software.* To appear.

[13] J. Claerbout. Reproducible Computational Research: A history of hurdles, mostly overcome. http://sepwww.stanford.edu/data/media/public/sep/jon/reproducible.html.

[14] J. Claerbout and M. Karrenbach. Electronic Documents Give Reproducible Research a New Meaning. http://sepwww.stanford.edu/doku.php?id=sep:research:reproducible:seg92, 1992.

[15] B. Flemisch. Online Survey: Reproducibility in Computational Science and Engineering (CSE). SIAM-CSE@siam.org, 2013.

[16] Purdue Online Writing Lab. http://owl.english.purdue.edu/owl/resource/560/10/, 2013.

[17] J.-L. Starck, E. J. Candès, and D. L. Donoho. The curvelet transform for image denoising. *Image Processing, IEEE Transactions on*, 11(6):670–684, 2002.

[18] Workshop on Sustainable Software for Science: Practice and Experiences. http://wssspe.researchcomputing.org.uk/, 2013.

Acknowledgments. The author thanks David Donoho and Ron Yang, from whom she was introduced to the key principles and best practices of reproducible research and robust enterprise software development, respectively. She also thanks Fred Hickernell for co-developing the definition of SSS and the principles of RRR via SSS during the development of GAIL [10].