Computational Science

“Same Old Silence, Same Old Mistakes”
“Something More Is Needed ...”

James J. Quirk
Los Alamos National Laboratory, quirk@lanl.gov

Today it is fashionable to portray computation as the third leg of science, the other legs being the classical disciplines of experiment and theory. But in the rush to promote computational science’s strengths, a blind eye is often turned to its weaknesses. This paper aims to increase awareness of a number of key deficiencies in the hope that the community can galvanize itself and tackle the identified issues head on. Specifically, the thesis to be developed here is that software automation could be used to package worked examples — in the form of dynamic electronic documents — that would allow interested parties, from different backgrounds, to communicate more effectively than at present. The hope, by making work easily repeatable, is that practical expertise can be properly archived. Currently, many avoidable mistakes are repeated time and time again as the mistakes do not lend themselves for journal publication and so go unrecorded.

1 Mission Impossible

The subtitle — “Same Old Silence, Same Old Mistakes” — is taken from a newspaper article[Col97] that highlights the billions, upon billions, of dollars wasted each year on failed software projects. The article draws insightful analogies between software failures and engineering disasters and it argues the case for more openness in owning up to software mistakes. Although written for a general audience, the article offers serious food for thought for the computational science community, especially now that many algorithms rely more on clever programming than on sophisticated mathematics. A reader of the article, G. N. G. Tingey, sums the situation up best when he writes, in a follow-up letter: the usual reasons for classical engineering failures are ignorance, arrogance and pride, including the shooting of the bearer of bad tidings. In the early part of the 21st Century, computational science appears to be failing, to some degree, on all these counts.

The challenge for this author is to rise above the anecdotal, with substantive evidence of community failings. Otherwise, you will be within your rights to serve
charges of ignorance and arrogance. In many ways this represents a mission impossible as it is notoriously difficult to get technical experts to agree to anything, en masse. A case in point at this adaptive mesh refinement (AMR) workshop was the willingness with which delegates criticized smooth particle hydrodynamics[Mon92] while engaging in circular arguments regarding the merits of patch-based AMR[BC89] versus cell-based AMR[PRQ92]. The view adopted here is that such arguments are moot, for algorithmic advantages depend on the actual application and they can be undermined by implementation details, as well as operator error.

Like mainstream society, computational science does not have a good grasp on software engineering. If it did, by now the idea of a standalone simulation code would be quite dead. Instead there would be sufficient standardization of operation that users could, if they so wished, run multiple algorithms on the same problem, from within some universal run-time system. This would allow rigorous head-to-head comparisons to be made for minimal effort and it would soon become clear which schemes performed well on which problems. Of course, the practicality of such a system lies in its many details.

Moving swiftly to the substantive elements of this paper. Section 2 revisits a blast wave simulation from 1991[Qui91] to show how increases in computing power, over the last decade, warrant a rethinking of how computational business is done. This theme is developed in Sect. 3 with an examination of the American Institute of Aeronautics and Astronautics (AIAA) policy on numerical accuracy[AIAA]. The policy was drafted in response to considerable concern with the quality of published numerical results and it is a solid attempt to raise computational standards. But in the spirit of this paper, the most illuminating aspect of the AIAA policy is the absence of any recognition that software could help power standards.

The aerospace industry is a mature discipline which, over the years, has been shaped by engineering failures, such as the ill-fated Comet[RAE54] and the Challenger disaster[RPC86]. Thus the AIAA policy can be thought of as raising the concept of computational technique to complement established practices like wind tunnel technique[PH52]. The usual difficulty, however, of defining technique is that it is a catch-all for all manner of hard-won, tricks-of-the-trade. But in the case of computations, Sect. 4 shows that software technologies do exist that would allow sound technique to be demonstrated using automated, worked examples. Again, obstacles to progress include software ignorance, software arrogance and software pride. Sect. 5 covers these sensitive issues from the perspective of the DOE Advanced Computational Software Collection[DM03].

Section 6 concludes with a set of recommendations. The way forward, essentially hinges on recognizing that software tools and management can be used to facilitate communication. And improved communication is needed for computational science to fulfill its potential.