A Methodology for Testing Mathematical Special Functions

The most important characteristic of software for special functions is *numerical accuracy*. As a first example of the need for testing, we look to Lozier and Olver, *Review of Mathematical Function Library for Microsoft Fortran, John Wiley and Sons, 1989*, NISTIR 4490, December 1990, in which it was found that the subroutine for evaluating the Bessel function $J_\nu(x)$, x and ν real, returns highly inaccurate or even totally incorrect values for certain arguments and orders. Thus the sign and all the digits of the computed value were wrong when $x \approx 9\pi$ and $\nu = 2\frac{1}{2}(2)10\frac{1}{2}$. The reason appears to be, at least in part, that an inappropriate normalization of a standard computing procedure, the Miller algorithm, was used. The algorithm was normalized by the computed value of $J_{1/2}(9\pi)$. Since $J_{1/2}(9\pi) = 0$, this cannot succeed. An alternative normalization based on equation (9.1.87) of Abramowitz and Stegun, eds., *Handbook of Mathematical Functions*, NBS Applied Mathematics Series 55, June 1964, would have avoided the failure.

There are two principal approaches to testing. The first is comparison against a standard. The second is verification of functional identities. *Comparison testing* is conceptually simple but requires computing in higher precision. *Verification testing* is performed entirely in one precision but requires great care in choosing an appropriate identity and applying the method because of the need to separate the error that arises from evaluating the identity from the error in the numerical evaluation of the function itself.

Both approaches require procedures for selecting test arguments and measuring error. Verification testing imposes an additional difficulty: the function must usually be evaluated at more than one point, and this leads to a process known as "purification" in which the arguments are perturbed after initial generation to help control the error that comes from evaluating the identity.

At the present stage of development, accuracy testing is usually limited to running cursory tests at the time when library software is installed. Since only a tiny fraction of the whole set of possible inputs is tested, the results cannot guarantee that the software will satisfy the needs of a particular application. Therefore, a need exists for improved software that can be used to characterize the numerical accuracy of special functions in any subset of the input domain, to any degree of precision.

A testing methodology that can be used to characterize numerical accuracy in this expanded sense should be oriented toward users and software analysts who are independent of the original software developers, and so the conceptual simplicity of the comparison method seems particularly attractive. Fortunately, computational obstacles that were burdensome in the past can be largely overcome today. Using the Internet, the World Wide Web, networks of workstations, and multiple precision computing techniques, it is possible now to establish, a *Software Testing Service for Special Functions* that will offer customized accuracy testing of special functions upon request.

As an example of how this service would be used, consider two software packages for the Airy function $\mathrm{Ai}(z)$ where z is complex. Package A is that of D. E. Amos, *Algorithm 644: A Portable Package for Bessel Functions of a Complex Argument and Nonnegative Order*, ACM Transactions on Mathematical Software **12** (1986), pp. 276–273. Package B uses the algorithm described in Lozier and Olver, *Airy*

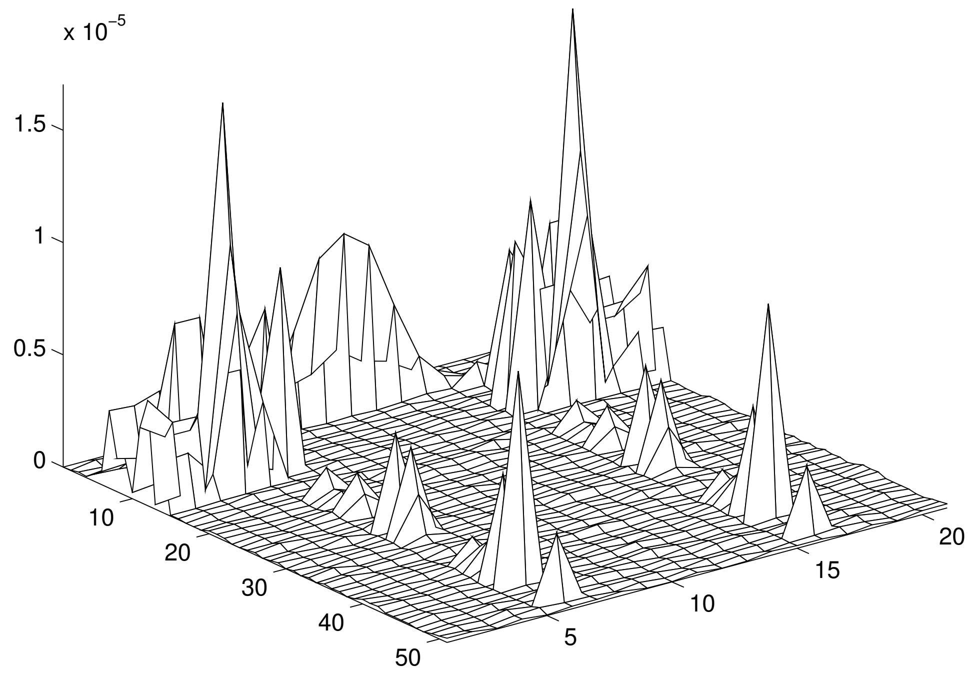

FIGURE 1. Error in D. E. Amos's software for the Airy function $e^{\zeta}\,\mathrm{Ai}(z)$, z complex, $\zeta = (2/3)z^{3/2}$, on the 51×21 grid with $|z_{ij}| = (i-1)/2$, $i = 1, 2, \ldots, 51$ and $\arg z_{ij} = -2\pi/3 + 2\pi(j-1)/30$, $j = 1, 2, \ldots, 21$. The maximum error on the grid is 1.7×10^{-5}.

and Bessel Functions by Parallel Integration of ODEs, Proceedings, Sixth SIAM Conference on Parallel Processing for Scientific Computing, vol.2 (R. F. Sincovec et al., eds.), SIAM, Philadelphia, 1993, pp. 531–538. Both packages provide an option to evaluate the scaled function $\overline{\mathrm{Ai}}(z) = e^{\zeta}\,\mathrm{Ai}(z)$ where $\zeta = (2/3)z^{3/2}$. Suppose we wish to choose between the two packages for computing $\overline{\mathrm{Ai}}(z)$ in the sector of the complex plane $|z| \leq 25$, $|\arg z| \leq 2\pi/3$.

Figures 1 and 2 show the results for Packages A and B, respectively. The figures are surface plots of the relative error in comparison to the double-precision version of Package A. All else being equal, we might choose Package B because the error appears to be more regular and predictable. The figures might also be of assistance to a software analyst in explaining the apparent defects in Package A.

Development of the testing service is progressing beyond the conceptual stage. A prototype software system will have several elements. First, an interactive interface is needed to present appropriate Web documents that will allow a user to specify test arguments and obtain test results. The user will also be required to specify certain arithmetic characteristics of his or her computer. The interface will be coded using established techniques, except possibly where more specialized techniques are required to handle very large data files. Second, software is needed to generate function values to the precision that is appropriate for the user's com-

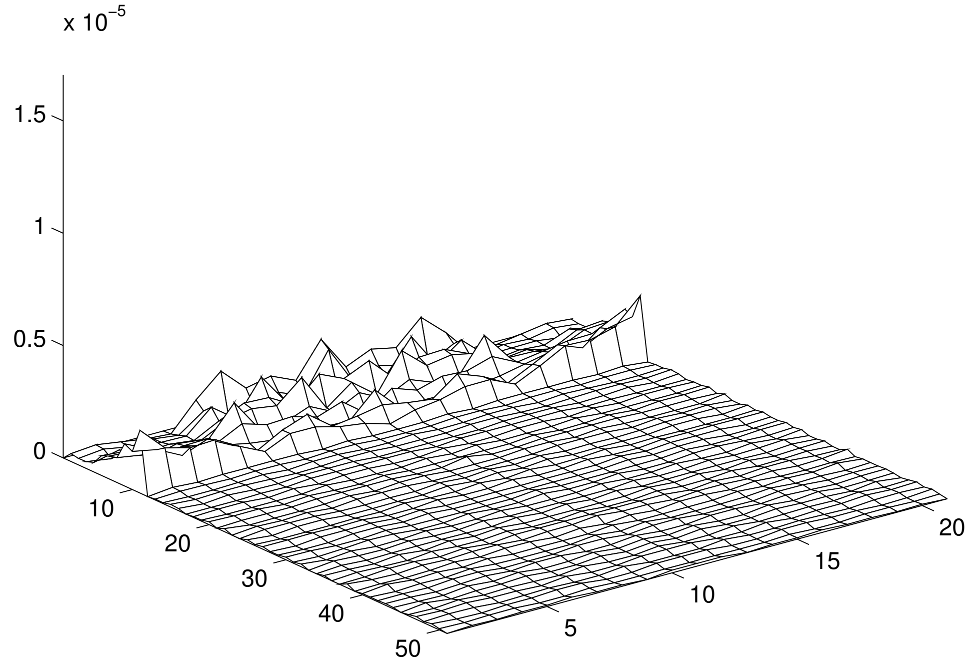

FIGURE 2. Error in a new algorithm for the Airy function $e^{\zeta}\,\mathrm{Ai}(z)$, z complex, $\zeta = (2/3)z^{3/2}$, on the 51×21 grid with $|z_{ij}| = (i-1)/2$, $i = 1, 2, \ldots, 51$ and $\arg z_{ij} = -2\pi/3 + 2\pi(j-1)/30$, $j = 1, 2, \ldots, 21$. The maximum error on the grid is 0.28×10^{-5}.

puter system. In the short term it is hoped that the extensive coverage of special functions in the Mathematica software system will be suitable for this purpose; see S. Wolfram, *Mathematica, A System for Doing Mathematics by Computer*, second ed., Addison-Wesley Publishing Company, 1991. The software package that is described in D. H. Bailey, *Algorithm 719: Multiprecision Translation and Execution of Fortran Programs*, ACM Transactions on Mathematical Software **19** (1993), pp. 288–319 is preferable because it is in the public domain, and so from the beginning of the project the user interface will communicate directly with Bailey's package, not Mathematica. Eventually high precision subroutines for mathematical special functions will be developed within Bailey's software package, a task that will require development of new algorithms and computer subroutines. Finally, a procedure is needed for distributing the task of function value generation to more than one computing facility. Since the planned testing service will allow arbitrarily large test requests, the computational burden may overtax any single computer, even if it is dedicated to the task. Since there are no data dependencies between successive calls to a special function routine, the testing service is exceptionally well suited to make use of existing techniques and software for managing the distribution and parallel execution of computer programs.