cFunctionComments

Introduction

This file provides documentation for the cFunctions.

Excel finds applications in a wide variety of disciplines, including business, finance, the sciences, and statistics. With such a broad constituency, it is hardly surprising that it is not serving all these different disciplines equally well. Specifically, Excel has been criticized in the statistical community for the sloppiness of many of its statistical functions and macros. Moreover, until 2003, Microsoft had not responded to this criticism by correcting any of those functions, even in cases where it acknowledged the problem and the validity of suggested solutions. In 2003, Excel fixed a number of these problems, specifically those around the standard deviation, the least squares routine LinEst, and the random number generator Rand; it has not fixed any of the remaining ones in 2007. Here we will only consider the latter, i.e., those errors that are common to, say, Excel 2000, Excel 2003, and Excel 2007. Any cFunction will be removed from out listing once Excel corrects its corresponding function.

To keep matters in perspective: Excel's non-optimal functions and macros are not defective in the sense that they will often fail under ordinary circumstances. In many cases, they may be perfectly adequate to their task. However, they are sometimes sloppily written, which may lead to an unnecessarily limited number of significant digits in their results. However, because spreadsheets are used in so many different contexts, they should be as accurate as possible just in case it matters for a particular application, especially since they do not inform their users as to how many of the digits shown are reliable.

Several decades ago, a compromise may well have been needed between accuracy and execution speed, but with contemporary computers that should no longer be a consideration. Moreover, in many cases, rather simple code corrections can improve the performance of spreadsheet functions, and defuse the criticisms leveled against them. The purpose of the cFunctions is to identify such existing problems in Excel functions and, where possible, to provide corrected versions, labeled with the prefix ***c***, that can readily replace the non-optimal ones in Excel. Do not confuse this with the C used to identify conversion, as in CBool, CCur, CDate, CDbl, CInt, CLng, CSgn, CStr, or CVar.

The cFunctions are offered here without charge, and without any warranty whatsoever. They are fully transparent, since their code is documented below, as well as the reference code used to test them. Leonardo Volpi's powerful Xnumbers.dll (http://digilander.libero.it/foxes/SoftwareDownloads) provides the convenience of an internal reference tool, so that we do not have to rely exclusively on external software. The Xnumbers verification codes are in the downloadable file xcFunctions.

Users are cordially invited to contact the author at his website, rdelevie@bowdoin.edu, with comments and suggestions for improvements or, better yet, with samples of additional cFunctions. It helps far more to remedy the situation for our own benefit then to lament the lethargy of Microsoft. Again, contributed cFunctions are most welcome, and will be tested and, if accepted, their authors fully acknowledged.

Organization

The functions to be tested here will be restricted to those for which problems have been reported or can be documented. This is still very much a work in progress. It started as an outgrowth of revising my book Advanced Excel for scientific data analysis for its a second edition, but I hope that, especially with help from readers, it will take on a life of its own.

In practice, the aim of cFunctions is to have a relative accuracy of at least 14 decimals, with the fifteenth digit playing the role of a guard digit. As our quantitative measure we will use the compact notation p*E*, where p stands for the negative 10-based logarithm (as in pH), and *E* for the error. In general we will use the *relative* error E_{rel}, except when that is impossible, in which case the *absolute* error E_{abs} will be used instead. Specifically, p*E* will be defined as

$$\mathrm{p}E = \quad -\log|f| \qquad \text{if } f \neq 0, f_{ref} = 0$$

$$\mathrm{p}E = \quad -\left|\frac{f - f_{ref}}{f_{ref}}\right| \qquad \text{if } f - f_{ref} \neq 0, f_{ref} \neq 0$$

$$\mathrm{p}E = \mathrm{p}E_{max} \qquad \text{if } f - f_{ref} = 0, f_{ref} \neq 0$$

$$\mathrm{p}E_{min} = 0 \qquad \mathrm{p}E_{max} = 14$$

There is little useful information in specifying negative values for p*E*, since p*E* = 0 already indicates that even the most significant decimal shown is unreliable. Both Excel and the National Institutes for Standards and Technology (NIST) Standard Reference Data (StRD) values provide only 15 decimals. However, different (and often unspecified) rounding conventions may yield results differing by ±1 in the fifteenth decimal, and we will therefore adopt the slightly more lenient convention of limiting the p*E* range from 0 to 14.

It is often useful to have a single number as an over-all criterion. An ideal function would have p*E* = 14 over its entire parameter space. A function with a much lower value of p*E* outside its range of applicability may still be quite adequate for that specific application. However, since we cannot predict what uses will be made of the spreadsheet, all our cFunctions are held to the requirement that they provide p*E* = 14 for all possible, valid input parameter values.

In the above definition, the reference values need to be sufficiently accurate. The references used in the present collection are based on using xnumbers.dll with a numerical precision of 200 decimal places. While use of xnumbers.dll makes cancellation and truncation errors in the reference values unlikely, it still runs the risk of introducing errors through the possible use of faulty algorithms. This risk is minimized by comparison with tabulated values in reference works whenever these are available. The specific programs used for computing the reference values are listed in the freely downloadable file cFunctionValidation, and readers are invited to scrutinize them, and to suggest improvements if they find any of these programs inadequate for their tasks. Patches for deficient Excel functions are provided in the collection of freely downloadable, ***c***orrected cFunctions.

Comments on the individual functions

The comments made here refer to the published versions Excel 2000 and Excel 2003 for pc's. There appear to have been no significant changes in these functions between Excel 95 and Excel 2000, or between Excel 2003 and Excel 2007. Suggested patches on the Microsoft Knowledge Base are not tested here, because many users may be unaware of their existence; they will be fair game for testing as soon as they are incorporated into future versions of Excel.

For functions of one or multiple parameters we will denote the argument by x, or by x_1, x_2, etc. For functions of one or more arrays we will use the argument y (the last letter of arra***y***) instead. The mathematical function names will be shown in lower case, their Excel implementations in caps, and the corrected Excel function as a lower-case c followed by the Excel name in caps.

Abbreviated reference is made to the following sources:

A&S: M. Abramowitz & I. Stegun, eds., *Handbook of mathematical functions*, NBS 1964, Dover 1965. Specific expressions are indicated by three period-separated numbers for chapter, verse, and equation number.

Atlas: K. B. Oldham, J. C. Myland & J. Spanier, *An atlas of functions*, 2nd ed., Springer 2008. Very useful compilation of expressions, often even more extensive than A&S. The accompanying **Equator** software is useful for checking numerical values.

CISE: H. R. Cook, M. G. Cox. M. P. Dainton & P. M. Harris, *Testing of spreadsheets and other packages used in metrology: Testing the intrinsic functions of Excel*, Center for Information Systems Engineering, Sept. 1999, *NPL report CISE* 27/99 downloadable from http://www.npl.co.uk/ssfm/download/documents/cise 27_99.pdf.

Heiser: D. A. Heiser, *Microsoft Excel 2000, 2003 and 2007 faults, problems, workarounds and fixes*, http://www.daheiser.info/excel/frontpage.html. The reference is followed by chapter number (or note letter), plus (where available) page number. Note that this document is updated occasionally, so that its page numbering may change.

ASINH(x)

(A note on nomenclature: *arc* in *arc*sin comes from *arc*us, i.e., curve or angle, whereas *ar* in *ar*sinh derives from *ar*ea. Excel and VBA abbreviate both prefixes to A; AS incorrectly uses arc even when ar is meant.)

An often quoted formula for the inverse hyperbolic sine is

$$\text{arsinh}(x) = \ln[x + \sqrt{(x^2+1)}] \qquad \text{(ASINH.1)}$$

with the symmetry property

$$\text{arsinh}(-x) = -\,\text{arsinh}(x) \qquad \text{(ASINH.2)}$$

When $x < 0$, the terms x and $\sqrt{(x^2+1)}$ have opposite signs, while their magnitudes for large values of x are almost identical. Even so, Excel uses (ASINH.1) to calculate its function ASINH(x) for all x. As a result, for $x < 0$, the Excel result can be quite inaccurate: for $x = -10^7$, Excel finds arsinh(x) = –16.805, whereas the correct value is –16.811;

for $x = -10^8$, Excel fails completely, because it first computes $x + \sqrt{(x^2+1)}$ as zero and then tries to take its logarithm, whereas the correct value is –19.114.

But the problems do not stop here because, even for positive arguments x, the Excel function fails at both small and large values of x. For small positive x, the root $\sqrt{(x^2+1)}$ becomes 1 when $x < 10^{-8}$, so that (ASINH.1) then approaches $x \to 0$ as $\ln(1+x) \approx x - x^2/2 + x^3/3 - x^4/4 + \ldots$, whereas the correct series expansion is (AS 4.6.31)

$$\operatorname{arsinh}(x) = x - \frac{1}{2}\times\frac{x^3}{3} + \frac{1\cdot 3}{2\cdot 4}\times\frac{x^5}{5} - \frac{1\cdot 3\cdot 5}{2\cdot 4\cdot 6}\times\frac{x^7}{7} + \cdots \qquad \text{(ASINH.3)}$$

The use of (ASINH.1) likewise affects the upper limit, because the computation becomes impossible once x^2 (rather than x itself) exceeds about 2×10^{308}. Indeed, the Excel function ASINH(x) fails above about 10^{154} even though, at that point, arsinh(x) is only about 355.3, not at all a very large number. But at $x > 10^{154}$ we have $(x^2+1) \approx x^2$ so that (ASINH.1) reduces to

$$\operatorname{arsinh}(x) \approx \ln(2x) \qquad \text{(ASINH.4)}$$

All the above considerations are taken into account in the corrected function cASINH; its behavior for small positive x-values is illustrated in Fig. ASINH-1.

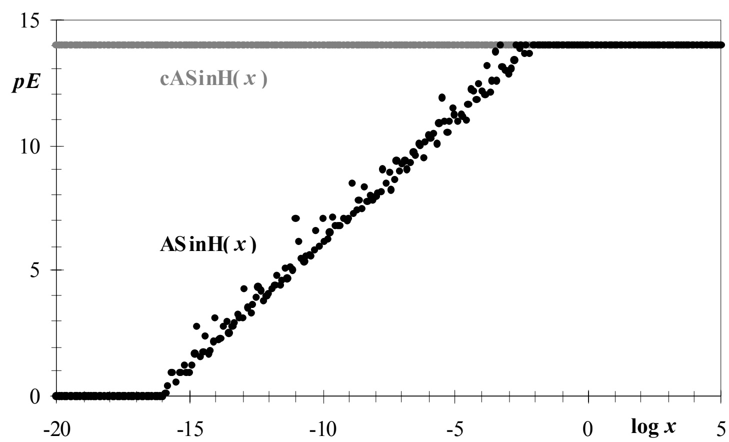

Fig. ASINH-1: Error plot of ASINH(x) and the corrected function cASINH(x), here both shown for the limited range $10^{-20} \le x \le 10^5$. The Excel function ASINH(x) remains at pE = 14 for $10^{-2} \le x \le 1.34\times10^{154}$, while the cFunction cASINH(x) yields pE = 14 over the entire range, $-1.8\times10^{308} \le x \le +1.8\times10^{308}$.

Erf (x)

Somewhat confusingly, Excel provides an unusual version of the error function in its Data Analysis Toolpak, with two arguments,

$$\operatorname{ERF}(a,b) = \frac{2}{\sqrt{\pi}}\int_a^b e^{-z^2}\,dz \qquad \text{(ERF.1)}$$

with a default value of 0 for the lower integration limit a. The Excel function has the restriction that neither a nor b can be negative, whereas the value of erf(a,b) will be negative when $b < a$. The Excel function gives an error message when a, b or both exceed 27.

The usual definition (AS 7.1.1)

$$\operatorname{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-z^2} dz \qquad \text{(ERF.2)}$$

has only a single argument, and is equivalent to setting a in (Erf.1) equal to 0, the default value in the Excel function. Below we will only consider the error function as defined in (Erf.2); the Excel function Erf(a,b) is equivalent to the difference erf(b) – erf(a).

The error function has the symmetry property (AS 7.1.9)

$$\operatorname{erf}(-x) = -\operatorname{erf}(x) \qquad \text{(ERF.3)}$$

and obviously allows negative arguments, which Excel's Erf(x) does only in a very limited sense.

Unfortunately, even when used as a single-argument function, i.e., with $a = 0$, and with a value for b between 0 and 27, the Excel function Erf(b) often has a rather low accuracy, and exhibits a number of discontinuities, as was already reported by Heiser. This erratic behavior, shown in Fig. Erf-1, is absolutely unnecessary: the non-alternating series (7.1.6) in Abramowitz & Stegun is converging smoothly over the range $-6 < x < 6$ while, outside that range, erf(x) = 1 to within 15 decimal places. This series has therefore been used for both the corrected function cErf(x) and the reference macro xErf().

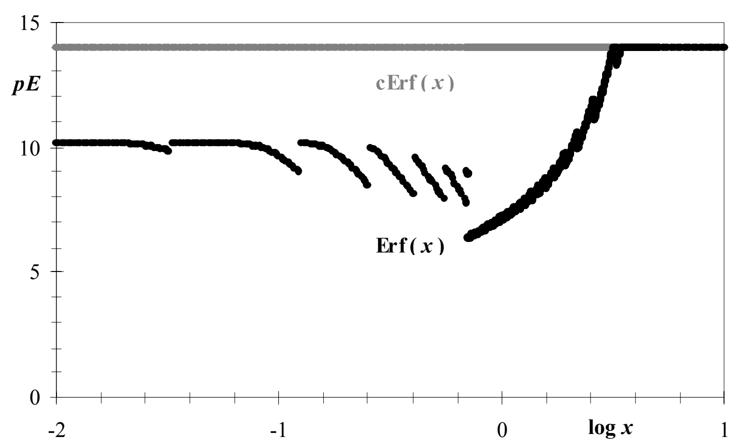

Fig. Erf-1: Error plot for Erf(x) and cErf(x) for $10^{-2} \le x \le 10$. Erf(x) provides answers for $0 \le x \le 27$. Outside the range shown, Erf(x) has a pE of about 10 for $0 < x \le 10^{-2}$, possibly because an incorrect value is used for the constant $2/\sqrt{\pi}$. Only for $10 < x \le 27$ does Erf(x) yield the correct answer, 1, with pE = 14. The corrected function cErf(x) gives results over the entire parameter range $-10^{308} \le x \le +10^{308}$ with pE = 14.

The corrected function cErf(x) can be used for any real argument, regardless of its sign or magnitude. Testing cErf(x) with xErf() yields agreement to within ±1 in the least significant digit, hence cErf(x) is rated as pE = 14, consistent with spot checks with Equator, which uses a quite different algorithm. For further details see A&S chapter 7, and chapter 40 in the Atlas.