Paper 2817-2018

NUMERIC PRECISION IN EXCEL AND SAS®: SIMILAR PROBLEM, DIFFERENT SOLUTION

Joe Matise, NORC at the University of Chicago

ABSTRACT

Numeric precision can be a difficult concept to understand for programmers, novice and advanced alike. This is compounded by the fact that not every software handles precision identically - even when using the same standard.
In this paper, the difference between how SAS® and Excel handle numeric precision on the boundaries is analyzed, and some strategies for handling the different issues that arise are suggested.
This paper is appropriate for any level of SAS, and requires no prior understanding of any of its concepts.

INTRODUCTION

Numeric precision refers to the number of digits a number can be exactly represented by. On machines, numbers are not stored with infinite precision; the number of bytes a number is stored in determines the number of digits that can be precisely represented.

In most modern computers, including Windows and UNIX machines, the IEEE 754 standard defines how a floating point number is stored; that is, a number that can have any number of digits before or after the decimal point, within the limit of machine precision. This is as opposed to an integer (a number with no digits after the decimal, or a "whole" number), or a fixed decimal number (which has a specified number of digits before and after the decimal). In SAS, numeric variables are all floating point numbers. SAS specifically uses double precision floating point numbers, which are stored in 8 bytes, by default. 8 bytes means 64 bits (each byte having 8 bits, or 1/0 digits). The IEEE 754 standard specifies that those 64 bits are allocated like so: 1 bit for the sign (positive/negative), 11 bits for the exponent (the scale of the number), and 52 bits for the mantissa (the precision of the number).

The largest integer that can be represented as a floating point number, thus, is 2^53, or 9007199254740992. Every number below that can be stored exactly using the mantissa; 2^53 itself is stored using the exponent, not the mantissa.

These numbers are slightly different on other systems, such as IBM mainframes, but for the purposes of this paper only IEEE 754 compliant systems will be considered.

HOW SAS HANDLES NUMBERS EXCEEDING MAXIMUM PRECISION

When a SAS number exceeds the maximum precision possible, the least significant digits are lost. In a decimal number, for example, if we could only store 3 digits, if we tried to store "1983", the 3 would be lost. We would be left with "1980", since the exponent would preserve the digit places.

Since numbers are actually stored in binary, then, at the beginning only a single place is lost: 9007199254740993 cannot be stored, and so it is still stored as 9007199254740992, but 9007199254740994 can be. No warning is given if you exceed this limit; SAS will simply store the

number with as much precision as it can, and it will appear to be rounding the number a nearby number (generally the next lower multiple of two that is storeable in maximum precision).

This is explained in more detail in the SAS documentation article "Numerical Accuracy in SAS Software", linked below.

HOW EXCEL HANDLES NUMBERS EXCEEDING MAXIMUM PRECISION

Excel uses same IEEE 754 standard as SAS, but it handles them slightly differently. As explained in https://support.microsoft.com/en-us/help/78113/floating-point-arithmetic-may-give-inaccurate-results-in-excel, Excel uses 15 significant decimal digits. This means slightly less precision is available in Excel than in SAS, as most 16 digit numbers can be represented exactly. Entering the equation =2^52 into Excel (and adjusting the format to display all digits) yields the value 4503599627370500 in the cell.

That number should actually be 4503599627370496, and would be in SAS, but Excel rounds it to end in 500. Additionally, Excel rounds down at 5 and up at 6, as opposed to traditional rounding up at 5 and down at 4.

Further investigation reveals that Excel does not actually reduce the precision of the underlying number, despite displaying the rounded number. Using a formula similar to "=D12-4000000000000000", we can remove the leftmost digit and see the remainder; if the number is stored at full precision and displayed rounded, it will show 503599627370496, while if it is stored rounded it will show 503599627370500.

Number	Number – 4x10^15
4503599627370500	503599627370496

As is clear, it is stored at full precision. Additionally, testing by importing the spreadsheet into SAS shows that the number is stored with full precision.

A spreadsheet showing this in greater detail available on Github is linked in the Resources section.

CONCLUSION

SAS and Excel both store numbers with similar precision, but Excel displays numbers with only 15 significant digits, while retaining full precision in the underlying number. SAS, on the other hand, will show numbers with all available digits, sometimes leading to misleading results. Care should be taken when using numbers near the precision limit, particularly when combining two technologies such as Excel and SAS.

REFERENCES

SAS Institue Inc. "Numerical Accuracy in SAS Software"(*http://documentation.sas.com/?docsetId=lrcon&docsetTarget=p0ji1unv6thm0dn1gp4t01a1u0g6.htm&docsetVersion=9.4&locale=en)*. Retrieved on 3/5/2018.

Microsoft, Inc. "Floating Point Arithmetic May Give Inaccurate Results in Excel" (https://support.microsoft.com/en-us/help/78113/floating-point-arithmetic-may-give-inaccurate-results-in-excel). Retrieved on 3/5/2018.

RESOURCES

To see this effect in action, download the spreadsheet at https://github.com/snoopy369/SGF2018/blob/master/Excel%20Precision.xlsx (uploaded on 3/5/2018).

ACKNOWLEDGMENTS

Thanks to Roger De Angelis for posting the original question on SAS-L that led to this question "Serious precision flaw in Microsoft Excel"(https://listserv.uga.edu/cgi-bin/wa?A2=SAS-L;3515dc16.1710b , retrieved on 3/5/2018), and to Quentin McMullen for participating in the discussion.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Joe Matise
NORC at the University of Chicago
Matise.joe@gmail.com