Microsoft Excel: Number representation, precision, and accuracy

Douglas A. Kerr

Issue 4
April 28, 2024

ABSTRACT

It is widely said that in the spreadsheet program Microsoft Excel numbers are represented, and calculations done, to a precision of 15 significant (decimal) digits. That is essentially true, but does not tell the whole story.

It is tempting to expect that the accuracy of calculations and results will be commensurate with that precision. But not necessarily—some times not even close.

The reason is that numbers are actually represented in Excel in a binary floating-point form, with a precision of 53 bits. While that scheme can precisely represent integers (to a precision of 15 significant decimal digits) over a gigantic range, it cannot always precisely represent numbers with a fractional part, even such a harmless-seeming one as "1.1". The consequences may be unexpected (and troublesome) errors when doing calculations in Excel.

1 INTRODUCTION

1.1 Microsoft Excel

Microsoft Excel ("MS Excel", or just "Excel"), a part of the application suite Microsoft Office, is a highly capable "spreadsheet" program.

1.2 My version

The details in this article have been verified by extensive testing on the version of Excel used in this office, Version 11, often called "Excel 2003", and which in fact dates from that year.

The underlying information upon which this article is based, however, is stated by various authorities as applying to later and current versions of MS Excel.

2 BACKGROUND

2.1 Precision and accuracy

Precision in this context refers to "how finely" is a numeric value expressed. A*ccuracy* describes "how correct" is the representation of a numeric value. The two are often confused with each other. But they are distinct properties, although admittedly often interrelated. I will return to that matter a bit later.

2.2 Specification of precision

2.2.1 *Number of decimal places*

A common metric for quantifying the property of precision is *number of decimal places*.

For example, this expression of a value:

435.78

is said to be to a precision of two decimal places (which implies that it is to a precision of 0.01 unit).

This expression of a value:

28.156

is said to be to a precision of three decimal places (which implies that it is to a precision of 0.001 unit).

This expression of a value:

41.520

is also said to be to a precision of three decimal places (which implies that it is to a precision of 0.001 unit).

Note that the "relative precision" afforded with a certain number of decimal places (think of the precision as a fraction of the value itself) varies here greatly with the size of the value. For this value:

1216.01

the implied relative precision is about 0.0008%, while for this value, wit the same number of decimal places:

5.02

the implied relative precision is about 0.2%.

2.2.2 *Number of significant digits*

Another common metric for quantifying the property of precision is *number of significant digits*[1].

[1] This is often called "number of significant **figures**". I use "digits" as that is less ambiguous.

For example, this expression of a value:

52.95

is said to be to a precision of 4 significant digits.

This expression of a value:

15728

is said to be to a precision of 5 significant digits.

This expression of a value:

215.350

is said to be to a precision of 6 significant digits.

But for this expression of a value:

26700

we cannot be sure of the precision in terms of the number of significant digits. It might be only 3, but could be 4 or 5.

Note that the “relative precision” afforded with a certain number of significant digits varies here with the size of the value, but to a far lesser degree than for a number of decimal places specification.

For this value

1216.2

the implied relative precision is about 0.008%, while for this value (with the same number of significant digits):

0.92035

the implied relative precision is about 0.001%.

3 PRECISION AND ACCURACY COMPARED

In, for example, a digital electrical voltmeter, when operating in a certain range, gives a result to 0.01 volt, which we can reasonably consider to be the precision in that case. From that, we might naïvely expect its accuracy to be ±0.01 volt. But it might very well be that, in this range, its accuracy is stated by the manufacturer as ±0.03 volt.

On the other hand, when we leave the realm of physical measurement and think in terms of the abstract manipulation of numerical values, consider a hand calculator that, in a certain situation, displays the result to 0.01 unit, which is the precision of that representation.

Almost always, this tells us that the accuracy of its result is $\pm$0.01 unit.

Hold that thought.

4 THE FLOATING POINT CONCEPT

4.1 General

What is spoken of as *floating point representation* is a scheme in which a very large range of numerical values can be represented, with essentially a fixed relative precision, by a modest set of digits.

One familiar example used in much technical work is *scientific notation*. There we might state a numerical value this way:

2.3654×10^{-18}

That of course would be the value that would be written in "ordinary" decimal form as:

0.0000000000000000023654

Far less writing is needed for the scientific notation form.

And of course we typically see the Avogadro number presented as:

$6.02214076 \times 10^{23}$

rather than as:

602214076000000000000000

Again, far less writing is needed for the scientific notation form.

The primary advantage of this notation is that very small, or very large, values can be written in a fairly compact way.

Note that under that same scheme the very same value could be written as:

41.28×10^{5}

or as:

4128×10^{3}

or in an infinity of other completely equivalent ways.

4.2 Normalized scientific notation

In certain contexts, it is the practice to "normalize" scientific notation. That means that the left part of the representation (often called the *significand*) always has one and only one digit (which must be non-zero) to the left of the decimal point. The right part,10 to a certain power, will be chosen to interact with that to represent the value of interest.

Thus, this value:

0.0000483

would be written in normalized scientific notation as:

4.83×10^{-5}

rather than, for example, as:

483×10^{-7}

even though that represents the same value.

4.3 Precision

As with "regular" notation, we can have different numbers of significant digits involved. For example here:

4.83×10^{-5}

the value is presented to a precision of 3 significant digits, whereas here:

3.2875821×10^{-3}

the value is presented to a precision of 8 significant digits.

Note that there are cases in which the implied precision is unclear in regular notation; consider the earlier example, 26700. If we wish to state that value, and make it clear that this is intended to be to a precision of only 4 significant digits, we can do that explicitly by expressing the value as:

2.670×10^{4}

4.4 The name "floating point"

The name "floating point" comes from the outlook seen in this figure:

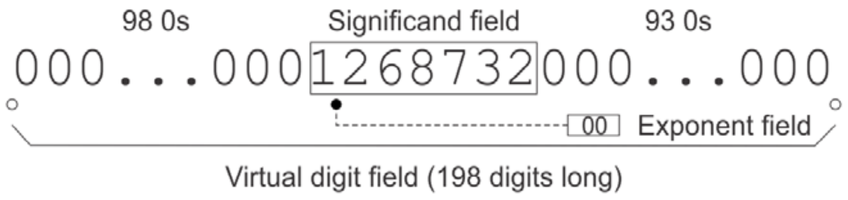

Figure 1.

Don't let my use of the word "field" suggest that this pertains to representation in a computer or such. This applies to the "human" representation I spoke of just above. I just use the term “field” for semantic clarity.

This figure is based on our using normalized scientific notation with 7 significant digits and a 2-digit “exponent” (power of 10) that can run from -99 to +99.

This outlook in effect creates a *virtual digit field* 198 digits long, with our 7-digit *significand* (which invariably contains the significant digits of the number being represented, thus its name) in about its middle, the rest of it being filled with 0s.

Its location of the decimal point is thus not fixed, but rather “floating”, thus the name of this scheme. The value in the *exponent* “field” effectively tells us where, in this 198 digit field, the decimal point is considered to be located.

In this example, the exponent is 00. That says the decimal point is to be considered to be where shown by the black dot. (The open dots, by the way, show the left and right limits of where the decimal point might be in this particular setup.)

This the number represented here is 1.268732, which we would write in normalized scientific notation as 1.268732 $\times$ 10^0.

In this figure:

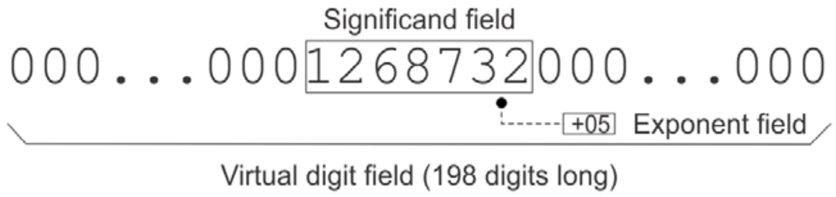

Figure 2.

the exponent is +05, which says that the decimal point is to be considered to be where shown. Thus the number represented here is 126873.2, which we could write in normalized scientific notation as 1.268732 $\times$ 10^{+5}.

In this figure:

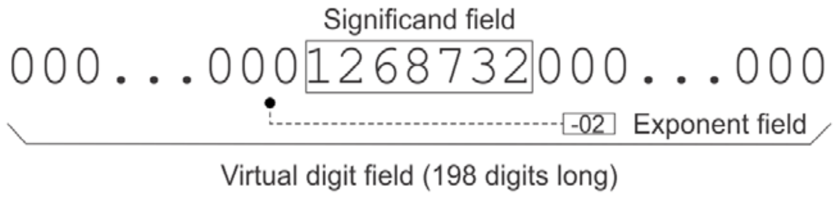

Figure 3.

the exponent is -02, which says that the decimal point is to be considered to be where shown. Thus the number represented here is 0.01268732., which we could write in normalized scientific notation as 1.2368732×10^{-2}.

We can easily imagine just this system being implemented in a computer, with perhaps 7 4-bit words for the 7 decimal digits of the significand and perhaps 2 4-bit words for the 2 decimal digits of the exponent. That would be a total of 36 bits for the representation.

Or perhaps we would go "full binary", with 24 bits to represent the significand. That would actually allow values up to 16.777215 to be represented as the significand, whereas only values up to 9.999999 have to be represented, but that is no harm.

We would still need 8 bits for the exponent, so it could represent any value in the range -99 through +99. (It actually would allow a greater range than that, which we would not exploit.)

Now, only 32 bits are needed for the representation. But that slight reduction in bit load is not a good enough reason to go "full binary". The main reason is that all the arithmetic in a computer is actually done on a binary basis, so all the numbers would have to be put into binary form anyway before any arithmetic operations could be performed.

5 IEEE 754

5.1 Introduction

In fact, in Excel, numbers are represented by a system much like what I just described. The system used is one defined by Standard 754 of the Institute of Electrical and Electronic Engineers (IEEE) [2], usually referenced as "IEEE 754".

In particular, it is the specific system officially identified as "Binary 64", but often called "double-precision binary".

In this plan, a number is represented in 64 bits, in terms of the following ingredients:

- A *sign* bit. (This of course tells the algebraic sign of the value, a matter that I will from here on ignore.)

[2] "IEEE Standard for Floating-Point Arithmetic"

- A *significand* field (52 bits), which gives the significant (binary) digits of the value.

  The significand is normalized, the most significant bit being just to the left of the "binary point". Since that bit will always be "1", it need not be encoded. Thus the number of bits in the significand is actually one greater than the size of the significand field (53 bits).

- An *exponent* field (11 bits), which gives the power of 2 by which the value indicated by the significand is multiplied to give the actual value.

  This is "biased" by -1023 so the range of exponents indicated runs from -1023 through +1024. But the exponent values at the extremes of that range are reserved to "flag" certain special situations (see Section 8).

There are many further details about this scheme, but they do not directly affect the story here here, so I will not present them.

6 NUMBERS WITH FRACTIONAL PARTS

6.1 Introduction

The IEEE 754 system of representation used in Excel serves us without further ado when the value of interest is (in decimal) an integer. Integers over a truly gigantic range can be represented without any oddities to a consistent precisions that we can think of as 15 significant digits.

But as soon as the decimal value involved has any fractional part, a gremlin can come to visit us.

6.2 Background—repeating fractions in decimal

We are quite familiar with the fact that, in the decimal realm, for a fractional part of a number such ⅓ or ⅐, the precise decimal representation has an infinite number of digits.

For example, this number: 25⅓ has this decimal representation:

$25.33333\underline{3}$

where the underscore tells us that the digit "3" repeats indefinitely.

As another example, the number 12⅐ has this decimal representation:

$12.\underline{142857}$

in which the digit pattern shown underscored repeats indefinitely.

6.3 Repeating fractions in binary

A similar thing can happen with the binary representation of a decimal value.

The tidy-looking decimal number 6.25 has this “tidy” binary representation:

110.01

But the tidy-looking decimal number 6.1 has this not-at-all tidy binary representation (to 23 significant bits):

$110.0001100110011001\underline{1001}$

where the underscored set of bits is repeated indefinitely.

That is, no representation in a finite number of binary digits can precisely represent the decimal value 6.1.

6.4 Implications in the IEEE 754 representation

Imagine that we start with the decimal number 1.0001. When that is converted into the IEEE 754 “Binary 64” form, in theory the significand (including the implicit leading bit) should be (to 100 significant bits):

1.000000000000011010001101101110001011101011000111000
10000110010110010100101011110100111100001101 1000_

where the underscore indicates that the significant bits actually go on forever.

But of course the significand field in that format has only 52 bits (not counting the implicit “1” at the beginning), so that value is in effect rounded to:

1.0000000000000110100011011011100010111010110001110001

That significand represents exactly this decimal number:

1.000099999999999988986587595718447118997573852539062 5

Although we started with a number that had only 5 significant decimal digits, it has now been transmogrified into one with 54 significant decimal digits! And with an error of about 0.00001. That is a relative error of about 0.01%, which is commensurate with a representation in only 6 significant decimal digits.

But Excel only wants to show us numbers on a “15 significant decimal digit” basis. So it rounds that number to 15 significant digits, giving:

1.00010000000000

This looks OK, seemingly the result we expected. But in fact, the value we saw above (all 54 significant digits of it) is the value that is still stored for this quantity in Excel. And it is not hard to think in terms of those 39 digits that were "rounded away" for viewing, but still actually present, "horning in" to some calculation, with surprising (and perhaps unpleasant) results.

We will see a specific example of that in Appendix A.

7 NUMBERS REPRESENTABLE IN THE IEEE 754 BINARY 64 FORMAT

Appendix B describes several classes of numbers that are precisely representable in the IEEE 754 Binary 64 format.

8 SPECIAL CREATURES IN IEEE 754

8.1 Introduction

The IEEE 754 standard provides for the explicit encoding of several special creatures. These matters do not directly impact the theme of this article, but for the sake of completeness regarding IEEE 754 I will give a concise discussion.

8.2 Zero

In a floating point format with an implicit first digit of "1" in the significand (as for the IEEE 754 format), there can be no direct representation of zero; the binary value will always contain at least one "1".

In IEEE 754, however, zero is arbitrarily represented by an exponent field of all 0s (not used otherwise) and a significand field of 0. The sign bit is still operative, allowing a distinction to be made between +0 and -0, a distinction that is meaningful in certain situations.

8.3 Infinity

Infinity is not an actual number, but in some work it is useful to be able to report that the result of a calculation would be "infinity".

In IEEE 754,Infinity is arbitrarily represented by an exponent field of all 1's and a significand field of 0. The sign bit is still operative, allowing a distinction to be made between +infinity and -infinity, a distinction that is meaningful in certain situations.

8.4 NaN ("not a number")

In some situations, it is useful to indicate for the value of some result (perhaps an invalid calculation) that it is "not a number" (abbreviated NaN).

In IEEE 754, NaN is arbitrarily represented by an exponent value of all 1's and a significant field of other than 0.

9 IN CONCLUSION

It is tempting to think that, if we do not require any precision in our Excel work better than 15 significant digits, the potential errors caused by the conversion of decimal numbers with fractional parts to and from binary form will not cause any difficulty. But, as we have seen, this matter can lead to surprising (and sometimes very troublesome) errors in our results. So—stay alert!

-#-

Appendix A
The gremlin at work

A.1 AN ACTUAL PROJECT

Here is an example of a relatively-dramatic error that occurred in recent actual work with Excel here.

I was preparing a discussion of the concept of limits in mathematics for a technical article in another field. I considered this equation:

$$y = \frac{(x+1)^2 - 1}{x} \qquad (1)$$

Of course, if x=0 this is undefined (since we would then have the accursed division by zero). But for very small x, y becomes very nearly 2. We say, "As x approaches 0, in the limit y=2".

In fact, we can see exactly how this happens by solving Equation 1 for y (assuming x≠0), from which we get

$$y = x + 2 \quad \text{for } x \neq 0 \qquad (2)$$

which is the exact value we should actually expect in each case (the second part of that tells us that the first part only applies if ≠0)

Nonetheless, I set up an Excel spreadsheet to demonstrate that this actually happens by evaluating Equation 1 for x=0.1, 0.01, 0.001, 0.0001, and so forth.

For x=0.00001, the actual value of y, from Equation 2, is exactly 2.00001. But the result given by Excel, evaluating Equation 1, is y=2.00001000001393 (note the 15 significant digits, as expected). This is indeed "very close to 2". But it is not the value to be expected from the solution to Equation 2, a hint that all is not well.

Farther down the table I have x=0.000000000001. This should give exactly=2.000000000001.

But for this, the result Excel gives, evaluating Equation 1, is y=2.00017780116468. This is not even close to the value we would expect from the solution to Equation 2.

Even farther down the table I have x=0.000000000000001. This should give y= 2.000000000000001.

But for this, the result Excel gives is y=2.22044604925031. This is very far from the value we would expect.

Note that these results far fall from the accuracy that we might think we would get from a program that represents numbers to a precision of 15 significant digits.

A.2 HOW DID THAT HAPPEN?

Consider the case where x = 0.00001. When we convert that value to binary, and add 1 to that, and square that, the result is (shown in decimal):

1.00002000010000*01392981175740715116262435913085937 5*

Since we think in terms of a precision of only 15 significant digits (shown underscored) the further digits (shown in italics) are seemingly "out of the picture"—we might think of them as "harmless junk". But they are not harmless—just staying "out of the battle zone" for now.

When we subtract 1 from that value, that process is precise, and so the result is (shown in decimal):

0.00002000010000*01392981175740715116262435913085937 5*

What are now the first 15 significant digits are shown underscored. But we see that now they include 5 "junk" digits (italic).

And this is how a significant error has crept into our result.

Finally, we divide that value by 0.00001. That operation is precise, so the result, shown in decimal, is:

2.00001000001392981175740715116262435913085937 5

Again I have underscored the 15 significant digits Excel will allow us to see, except it actually does this by rounding, not truncation, giving us

2.00001000001393

which is not at all the value:

2.00000000000000

we were expecting, and which is correct.

-#-

Appendix B
Numbers precisely-representable in the IEEE 754 Binary 64 format

Certain classes of numbers can be precisely represented in the IEEE 754 Binary 64 format, and thus presumably in Excel. I describe the perhaps most important ones below.

This is based on empirical observations made on a (presumably authentic) IEEE 754 calculator.

1. Any integer of up through 16 digits.
2. Any positive power of 10 up through 10^{22}.
3. Any positive or negative power of 2, or any sum of powers of 2, in the range of the format (see below).
4. The product of any value in item 2 and any value in item 3 (within the range of the format).

Overall range of the format

The largest number that can be represented is approximately 1.79769×10^{308}

The smallest number that can be represented is approximately 4.9406×10^{-324}

-#-