

III. EXCEL COMPUTATION AND DISPLAY ISSUES.....	1
WHY IS THIS SECTION IMPORTANT?.....	1
THE COMPUTING ENVIRONMENT.....	3
THE GENERAL DIFFERENCES FROM MATHEMATICS.....	3
THE VARIANT STRUCTURE IN EXCEL.....	3
CELL DATA ENTRY.....	4
THE EXCEL RANGE STRUCTURE.....	5
THE VALUE STRUCTURE IN STATISTICS, PSYCHOLOGY, EDUCATION AND SOCIAL SCIENCES.....	6
CHARACTERISTICS OF THE DATA COMMONLY USED.....	6
COMBINED NOMINAL AND ORDINAL DATA.....	7
FITTING THE DATA TO THE VARIANT TYPES.....	9
FLOATING POINT NUMBERS IN EXCEL.....	10
THE STRUCTURE.....	10
FLOATING POINT NUMBERS WITHIN EXCEL.....	11
ARITHMETIC ON FLOATING POINT NUMBERS.....	12
THE PERCEPTION OF NUMBERS.....	13
ZEROS, AND SMALL VALUES.....	14
THE IMPORTANCE OF CORRECTLY HANDLING AND COMPUTING SMALL NUMBERS.....	15
MAXIMUM VALUES.....	15
INHERENT FLOATING POINT LIMITS TO ACCURACY.....	15
SEQUENCES OF FLOATING POINT CALCULATIONS.....	16
DISPLAYS AND OUTPUT.....	16
DISPLAY OF THE {IF} OBJECT.....	16
EXCEL 2003 AND 2007 DEFAULT DISPLAYS.....	17
NON-NUMERIC ERROR MESSAGE RETURNS.....	19

### **III. EXCEL COMPUTATION AND DISPLAY ISSUES**

#### **WHY IS THIS SECTION IMPORTANT?**

Why is all this in section 3 important? It gives a road map of how things occur in Excel, how they appear to the user, and what are intrinsically, limits. These are limits on the accuracy of computations and limits on the ability to reduce “real world quantitative and

qualitative data” to “a statistic”. The flexibility of Excel (when used in all kinds of different computers and architectures) to display either qualitative or quantitative data in worksheet cells, and the flexibility in changing how it is displayed, leads to misinterpretations and incorrect conclusions about displayed cell contents.

The issues of actual errors and inaccuracies in the values displayed in cells and worksheets comes from at least seven conceptual levels:

1. Conversion of data, measurements, test results, survey results, money, dates, time, counts, verbal responses, classifications, assessments, indexes, etc into numbers or other mathematical/logical terms and symbols. These involve problems of relating theories and measurements to numbers that have the properties of integers or real numbers (which can be correctly ranked, added, subtracted, multiplied, divided, transformed, related by equations, etc.). They also include the problems of establishing logical symbolic structures. The error here is between the concept (and theories) and the resulting numbers, dates, times, textual terms and symbols.
2. Transfer of the information into an Excel spreadsheet. The error and inaccuracies are one of correct transcription (transfer).
3. Establishing mathematical or logical equations or arithmetical operations. The errors here are incorrect equations, incorrect cell references, incorrect functions, incorrect ranges, errors in naming and specifying ranges, incorrect array function usage, and all kinds of errors that can easily occur on an Excel worksheet. This is a user fault/error for which Excel can't be faulted, except when HELP gives poor, overly complex or erroneous information.
4. Uncertainties in theory or ability to represent concepts as unique mathematical representations. Includes those situations where theories differ or there are different equations (or relationships) to represent a concept or theory. Also includes areas where different approximating relationships (or equations) exist. Each of these different methods will give different results. If standard values exist, then the differences from the standard can be considered as errors.
5. Computer operations on the information in each cell. Differences will be observed that are from the peculiarities of the specific computer hardware. These come from the peculiarities of individual computer “chip-sets” (essentially differences in the way addition, subtraction, multiplication and division is done and implemented), the degree of conformance to industry standards (such as the IEEE-754 standard), details on how the higher level programming language (VBA, FORTRAN, or C or other) is converted to machine instructions (chip set instructions at the bit/byte/word levels) and then at the higher levels, the specific programming language statements. This area is problematic, since errors and faults can be identified, but they are not correctable. Most of the error comes from round off.
6. Faults in the programming language (VBA) algorithms.
7. Those that come from the fact that a computer only approximates solutions of mathematical equations.

# **THE COMPUTING ENVIRONMENT**

## **THE GENERAL DIFFERENCES FROM MATHEMATICS**

Statistics is usually introduced as a course in mathematics, by the math department. This leads to a primary view that the numbers and process of computation used in statistics follows mathematical theory and is not limited in any way by its representation by real numbers. The symbol  $\{\text{IR}\}$ <sup>1</sup> represents the traditional, statistical concept of a statistic or equation as a purely mathematical object, or field.

A binary computer operates on strings of bits that are designed to represent  $\{\text{IR}\}$  objects. These representations are defined as  $\{\text{IF}\}$  objects.  $\{\text{IF}\}$  objects are a finite subset of  $\{\text{IR}\}$ , in which the rules of mathematics are only approximated. In a computer, addition and multiplication of  $\{\text{IF}\}$  objects are not associative. The summation in  $\{\text{IF}\}$  is not well defined, and usually is taken as a number when its value no longer changes. This no-further-change limit is referred to as being  $\{\text{IF}\}$ -convergent, which is different from  $\{\text{IR}\}$ -convergent. The harmonic series (sum of  $1/i$ ) in  $\{\text{IR}\}$  is divergent, but in  $\{\text{IF}\}$ , it is  $\{\text{IF}\}$ -convergent. The  $\{\text{IF}\}$ -convergent value can be different, depending on how the internal algorithm does associations. The sum of integers is  $\{\text{IF}\}$ -convergent, because there is a limit on the size of integers that can be represented as  $\{\text{IF}\}$  objects. (Gentle 2004).

## **THE VARIANT STRUCTURE IN EXCEL**

Each cell on the worksheet is represented as a class of representations called a variant variable made up of 128 or more bits. There are 18 types identified within the variant structure. Of the 18, 7 represent  $\{\text{IF}\}$  numbers, 1 represents date/time, 1 represents strings of characters, 1 represents logic, and 2 represent other objects. There is a VarType constant associated with each one of the 18 types. The Help section in Visual Basic contains information about variant variables.

All computed numbers are a binary representation of a number. The structure can be in the form of an integer (16 bits), a long integer (32 bits), a single precision number (32 bits), a double precision number (64 bits), currency value (64), decimal value (112) and a byte value (8). The single precision, double precision, currency and decimal value numbers are all  $\{\text{IF}\}$  objects. Integers that remain within the size limitations can be considered  $\{\text{IR}\}$  numbers, since the arithmetic is exact. The Boolean variables of TRUE and FALSE are represented by integers, 16 one-bits in the integer form and 16-zero bits

---

<sup>1</sup> From Gentle (2004). "It is important for people who deal with numerical computations to understand that the computer works only with a subset of real numbers. I use the symbol (Word does not have it, so  $\{\text{IR}\}$  is used) to represent the real numbers. This is a special kind of mathematical object, a field. I use the symbol (Word does not have it, so  $\{\text{IF}\}$ ) to represent the object that the computer uses to simulate  $\{\text{IR}\}$ . These are called floating point numbers...The object defined by  $\{\text{IF}\}$  is a finite subset of  $\{\text{IR}\}$ , it is not, however, a field (.nor any other object that mathematicians commonly define and study). ... Second, while there are two basic operations in  $\{\text{IF}\}$ , just as there are in a field, these two basic operations (called addition and multiplication) are not always the same as the two corresponding operations in  $\{\text{IR}\}$ . Neither of these two operations is associative in  $\{\text{IF}\}$ ."

in the latter represent the Boolean variables of TRUE and FALSE. These are {IR} numbers.

The arithmetic of addition, subtraction, multiplication and division can be done only with integer, long integer, single-precision, double-precision and currency numbers. Currently, decimal variables cannot be declared in VBA routines. When the arithmetic includes logic variables, a FALSE is taken as a zero value and a TRUE is taken as a -1 value. When a count is made within a range (using the COUNTA function), logic values of TRUE are counted as +1, and FALSE values are counted as 0. Otherwise they are taken as zero. In {IF} arithmetic, a logical TRUE times a number (say =55) results in a minus value (i.e. -55), whereas in {IR} arithmetic, a logic TRUE value is one (plus one). When integer or long integer numbers are involved in division, the division only returns the integer portion of the result. Multiplications of integer and long integer values are limited by the assigned value limits of the product as signed integers. An error code will be returned when these limits are exceeded.

Normally all numbers in a cell are either a double precision number or a text (string) form of the number. Currency numbers can only be used in VBA routines, since the worksheet cell values cannot be preset to currency types. The currency format selected under Format – Cells – Currency just inserts the \$ symbol to a fixed point number. (e.g. the floating point number is converted to a fixed point character (text) display and the “\$” added.)

These facts clearly indicate that {IF} results may be quite different from {IR} expectations.

There is an add-in package (see Note XA) available for use as VBA functions, that exploits the variant cell structure. The multi-precision functions in essence use the character string variant to handle strings of digits (numbers) as characters. (This is similar to the way Excel handles complex numbers.) Using spreadsheet and VBA constructs and “xnumber46” functions, the full range of mathematical operations can be done on these strings of digits (up to 200 digits). However when the result (as a string) is entered into an Excel cell, a conversion to the standard Excel double precision form is necessary in order to do the normal Excel cell computations and displays. By using the “xnumber” functions in cell equations, the extended precision can be retained. Note XA describes how this is done, and how the “xnumbers” add-in can be loaded into Excel.

## **CELL DATA ENTRY**

When something is entered into a cell, Excel uses an internal parsing algorithm to find out which of the variant types the entry is. Coded identifiers (such as abbreviations) for variables may be misinterpreted as dates or numbers. This locks the cell into a given variant type, which persists with changes to the entry. De Levie (2005) has pointed this out as “hidden problems”. One has to at this point select the cell, select Edit from the tool menu, Edit → Clear → All to restore to the original unknown setting.

If the cell format is retained in the default (GENERAL) mode, then if the entered characters/number sequence starts (anchored to) from the LEFT cell boundary, Excel has determined that it is TEXT. If the character/number string builds from the RIGHT cell boundary (anchored to), Excel has determined that it is a NUMBER. Setting cell formats ahead of time will change how Excel interprets the input.

When a cell entry starts with the “=” sign, Excel expects an equation or function to be entered. The equation is “text” that normally will return a number, so that the number format is retained. The General format prevails if not changed. Therefore, both text and numbers can be correctly interpreted. There is a default here that exists for entry into the very first row of a worksheet. If the first row is set to text (normally column headings), then cells in the first row will not properly recognize equations (“=...”), even when the cell format is changed back to numbers.

If the cell format is set to numbers, then if by accident, a non-number (excluding signs, and the exponent E) is entered, the entry will revert to text.

Most Excel users are never clear on the difference between text and numbers as handled by the Excel program on data input. All keyboard inputs are characters, and each key generates an 8 bit binary string that goes to the computer. The ASCII convention identifies what are numbers, text characters (alphabet, symbols, foreign language characters, mathematical symbols, etc.) and controls. When keys or combinations of two or more keys are pressed, by mistake, the computer is confused by what to represent the entry by. The user is unaware that what he intended to enter, is not what actually got entered. So he says that the program is at fault, not his fingers. The change may not be evident from what is visually seen as cell contents.

Note G expands on the data input problem and recovery.

## **THE EXCEL RANGE STRUCTURE**

Excel functions and routines basically work with the input data in the form of ranges. The range can be a series of cells in a row, a series of cells in a column or a block of cells within the worksheet. A descriptor of the form “A1:C12” is a range, where in this case it is a contiguous block of 36 cells, starting at cell A1 and ending at cell C12. There are many statistical functions and Data Analysis routines that take ranges as inputs. This means that when one uses an Excel function such as LINEST, a contiguous range of input data must be built, by copy and paste efforts gathering the data from different locations into a single range block. KBA 214117 describes this situation and what workarounds must be done to get the functions to work. The separate within-ranges of each X variable, and the Y variable must not overlap either.

The important point here to remember is if the range is data (numbers) THERE MUST BE NO BLANK CELLS IN THE RANGE. There are however, functions such as COUNT and DEVSQ that will ignore blank cells. A simple test for blanks in a range is to input the equation in an outside cell, =ISBLANK(the range). If a 1 appears (true) then there is a blank in that range. The behavior of Excel is not consistent when a blank occurs in a range of input data.

Another important point is to be sure that if the range contains numbers, that there are no cells in the range that has text. There is an important exception here, when the top cell of a column of data has a name for the data, this is called a “label”. Many of the Data Analysis statistical routines allow the label to be included in the range, if the label box is checked on the input box. Most of Excel's functions will output an error code if a non-numeric entry is in the range.

# **THE VALUE STRUCTURE IN STATISTICS, PSYCHOLOGY, EDUCATION AND SOCIAL SCIENCES**

## **CHARACTERISTICS OF THE DATA COMMONLY USED**

Values in these sciences are based on measurement theory. "Measurement theory is a branch of applied mathematics that is useful in measurement and data analysis. The fundamental idea of measurement theory is that measurements are not the same as the attribute being measured. One has to recognize that here the process of measurement and analysis is in itself a social process that itself is shaped by human political forces. Very often these concepts cannot be expressed in {IF} form, and as a result, there is a gap between the concept and its representation". (Sarle 1996)

In general we can speak in terms of levels of measurement:

**Nominal:** Two things are assigned the same symbol if they have the same value of the attribute. Permissible transformations are any one-to-one or many-to-one transformation, although a many-to-one transformation loses information.

**Ordinal:** (Ordered categories) Things are assigned numbers such that the order of the numbers reflects an order relation defined on the attribute.

- Each category may be independent or represent a summation of categories of lower rank.
- Two things  $x$  and  $y$  with attribute values  $a(x)$  and  $a(y)$  are assigned numbers  $m(x)$  and  $m(y)$  such that if  $m(x) > m(y)$ , then  $a(x) > a(y)$ . Permissible transformations are any monotone increasing transformation, although a transformation that is not strictly increasing loses information.
- A problem here is the "width" of each category in relationship to the "widths" of adjacent categories. A large "width" creates large "tie" blocks.

**Interval:** Things are assigned numbers such that differences between the numbers reflect differences of the attribute.

- If  $m(x) - m(y) > m(u) - m(v)$ , then  $a(x) - a(y) > a(u) - a(v)$ .
- Permissible transformations are any affine transformation  $t(m) = c * m + d$ , where  $c$  and  $d$  are constants; another way of saying this is that the origin and unit of measurement are arbitrary.
- Requires a distinction between a few intervals or as many intervals as there are cases.
- Each interval may be independent or represent a summation of intervals of lower rank

**Log-interval:** Things are assigned numbers such that ratios between the numbers reflect ratios of the attribute.

- If  $m(x) / m(y) > m(u) / m(v)$ , then  $a(x) / a(y) > a(u) / a(v)$ .

- Permissible transformations are any power transformation  $t(m) = c * m^{** d}$ , where  $c$  and  $d$  are constants.

Ratio: Things are assigned numbers such that differences and ratios between the numbers reflect differences and ratios of the attribute.

- Permissible transformations are any linear (similarity) transformation  $t(m) = c * m$ , where  $c$  is a constant.
- Another way of saying this is that the unit of measurement is arbitrary.

Absolute: Things are assigned numbers such that all properties of the numbers reflect analogous properties of the attribute.

- The only permissible transformation is the identity transformation.
- Examples: number of children, the number on a football players jersey (Sarle 1996 with additions from other sources)

Most statistics textbooks combine Log-interval, Ratio and Absolute measurement levels into one term. Larson (2003) and Triola (2000) use the term ratio for the combined level, with the result that there are only 4 levels of measurement. Moore and McCabe (2003) say this is beyond introductory statistics, and only use two levels, categorical and quantitative. Others just use the terms qualitative and quantitative levels. At the two-level situation, nominal and ordinal are treated as categorical or qualitative data, and the rest as quantitative data.

### **COMBINED NOMINAL AND ORDINAL DATA**

“Observations on an ordinal variable represent responses to a set of ordered categories, such as a five-category Likert scale. It is only assumed that a person who selected one category has more of a characteristic than if he/she had chosen a lower category, but we do not know much more. Ordinal variables are not continuous variables and should not be treated as if they are. It is common practice to treat scores 1,2,3, ... assigned to categories as if they have metric properties but this is wrong. Ordinal variables do not have origins or units of measurements. Means, variances, and covariances of ordinal variables have no meaning. The only information we have are counts of cases in each cell of a multiway contingency table.” (Jöreskog 2002) Note J is a discussion among many teachers and researchers about how to deal with data in the form of Likert scale.

One common expression of ordinal data is shown in table 3-1.

**Table 3-1: Typical Ordinal Data**

Category	Jöreskog Symbols	Jöreskog Coding	Traditional Likert Values (Inverted)	Traditional Likert Values
Agree Strongly	AS	1	5	1
Agree	A	2	4	2
Neutral			3	3
Disagree	D	3	2	4
Disagree Strongly	DS	4	1	5
Don't Know	DK	8	Blank	Blank
No Answer	NA	9	Blank	Blank
Not Applicable			Blank	Blank

Studies have consistently shown that there is a difference in responses from surveys, depending on whether the questions are positively or negatively worded, and whether the first response is positive or negative. One has to add a binary or categorical variable to whether the question is “forward” or “backward” oriented, when the questions are mixed.

A common view is that somehow, one can interpret ordinal data on some kind of continuous scale, contrary to Jöreskog's view, which is mathematically correct. Searle says, “In real life, a scale of measurement may not correspond precisely to any of these levels of measurement. For example, there can be a mixture of nominal and ordinal information in a single scale, such as in questionnaires that have several non-response categories. It is common to have scales that lie somewhere between the ordinal and interval levels in that the measurements can be assumed to be a smooth monotone function of the attribute. For many subjective rating scales (such as the 'strongly agree,' 'agree,' ... 'strongly disagree' variety) it cannot be shown that the intervals between successive ratings are exactly equal, but with reasonable care and diagnostics it may be safe to say that no interval represents a difference more than two or three times greater than another interval.” (Sarle 1996)

Note J, includes a discussion from others on the issues and problems of handling Likert data as representing a continuous scale.

A constant, reoccurring problem with survey and questionnaire data is when “don't know, no response, no answer or no opinion” occurs. The “no response” cannot be translated to a number in the same sense that the survey-questionnaire items are on a “ranked” or numerical value scale. Putting in a zero for no-response or a zero for missing data severely biases the analytical results. Currently there is no mathematical structure in the {IR} sense that deals with this problem.

“The usual way to deal with such responses is to declare them as missing values. (Jöreskog 2002) Then the software program is used to artificially put in values (imputation), essentially negating the information in the response. Excel does not have any imputation routines. If the cell is left blank, then it is handled as a missing number, and handled differently by the different functions and routines.

Another way is to establish functional relationships (i.e. sets of equations) between the ordinal variable and one or more continuous variables in some theoretical model. For

example, the count of those who “agree strongly” on a statement “I don’t think that public officials care much about what people like me think” is the intersection of two ordinal measures. The intersection as a count becomes a new absolute data type. It can be converted to a continuous variable (a proportion) with this name and the variable expresses a proportion of the “population” with this view. It is just an extension of the contingency table concept. The equations then express how this measure is related to other measures (i.e. demographic and personal measures) in the form of regression like relationships. For example an equation can be formed relating the fraction of “no-answer” in the population to a specific view “I don’t think that public officials care much about what people like me think” based on gender, age, education level, income level, etc.

There have been academic efforts to build additional functions, routines and specific worksheets to analyze categorical data within Excel. The method of building contingency tables and obtaining p values or other continuous variable measures from column and row totals is within existing Excel capabilities.

Anyone who attempts to use questionnaires, surveys, expert assessments, etc to generate data, should be familiar with the literature. Some recommended books on this are listed under the section “Measurements in the Social Sciences” in the bibliography (Section 20).

### **FITTING THE DATA TO THE VARIANT TYPES**

Nominal data can be entered as string types. Computation is limited to simple logic of testing for equality or non-equality. Addition (+) of strings represents concatenation, where the strings are combined into a new string. The new string is not equal to either of the components. The symbols “-, \* and /” result in error returns. Accuracy is entirely dependent on the characters entered in the cell. Spaces, which are not visible, may be entered by mistake, and result in incorrect results. Counts of nominal data become quantitative measurements of the absolute type.

Ordinal data and some mixtures require the operators <, <=, >, >= besides the = and ≠ operators on the measures. Data can be entered as strings, where the first and second characters establish ranking, such that by sorting, the proper ranking order is established. A “value” between two ranked ordinal data items obtained by arithmetic, in a sense does not exist. The “value” only exists if it is one of the “items” on the ranked list. Counts of ordinal data become quantitative measurements of the absolute type.

If the data is entered as “text” or string type, then there is a definite way in which different cells are handled in terms of the operators <, <=, >, >=, = and ≠. The process is a conversion from left to right on the string, converting the characters to the corresponding ASCII number (0 to 127), and doing the operation on the resulting ASCII numbers, in left-right sequence. If the text string contains numbers, the resulting operations (including sorting) will result in unexpected behavior. For example “11 ” is determined to be less than “2 ”. Excel however does allow for a special sort that preserves the normal view of numbers and text. However this does not extend to the operators on individual cells.

Computing an average or a standard deviation implies that the data is at the interval or higher level, and can be represented by numbers.

The other measures can be treated as numbers, (integers or floating point) where arithmetic can be applied to a limited, or full extent, depending on the real life aspects of the measurements.

When this transformation is made to Excel variants, and arithmetic and functions on the values applied, the issue of accuracies may not even be relevant, or even applicable

## **FLOATING POINT NUMBERS IN EXCEL**

### **THE STRUCTURE**

95% of all Excel users are clueless about floating point<sup>1</sup>. This does not stop users from doing all kinds of computations under the assumption that every computation in Excel is exact. (after Kahan 2004)

From KBA 78113 (with additions and modifications), Cornea-Hasegan and Norin (1999) and

“A floating-point number is stored in binary in three parts within a n-bit range: the sign, the exponent, and the mantissa. There is a fourth part, an implied bit that does not have a position since it is an implied bit.

The rightmost bit is the 0 position, and the leftmost bit is the highest position.

“The sign stores the sign of the number (positive or negative), the exponent stores the power of 2 to which the number is raised or lowered (the maximum/minimum power of 2 based on a IEEE-754 bias standard), and the mantissa stores the actual number. The finite storage area for the mantissa limits how close two adjacent floating point numbers can be (that is, the precision).

A number  $n$  is expressed in floating point format as

$$n = s \cdot M \cdot N$$

where  $s$  is the value of the sign bit,  $M$  is the exponent field, and  $N$  is the significant field, where  $M$  and  $N$  are the number of binary bits in the field. The significant field ( $N$ ) is also referred to as the mantissa of the exponential representation of the number.

There are three standard representations, based on the IEEE-754 standard, on how floating point numbers are represented in a computer. The actual computations are done in the CPU. Consequently there is translation involved between the IEEE-754 representation of the number in computer memory and the representation in the CPU registers where the computations take place. Cornea-Hasegan and Norin (1999) describes how the computation takes place in the Intel IA-64 architecture (or ix86 type CPUs).

---

<sup>1</sup> The term “floating point” is a generic computer science term. It refers to a specific method of representing any number by means of an exponential notation.

### **FLOAT [Single Precision](Single){4 bytes}<sup>2</sup>:**

1 Sign Bit	8 Bit Exponent	1 Implied Bit	23 Bit Mantissa
------------	----------------	---------------	-----------------

The implied bit (1) is not transferred to memory, and therefore is not in the saved number. Early versions of Excel used single precision.

### **DOUBLE [Double Precision](Double){8 bytes}:**

This is the default representation for numbers in an Excel cell.

1 Sign Bit	11 Bit Exponent	1 Implied Bit	52 Bit Mantissa
------------	-----------------	---------------	-----------------

The implied bit is not transferred to/from memory, but inserted as part of the computer instruction.

### **LONG DOUBLE [INTEL Double Extended Precision]{10 Bytes}:**

This is the Intel IA-64 instruction set representation within the CPU (Intel x86 and clones), where the actual arithmetic operations are done.

1 Sign Bit	15 Bit Exponent	1 Explicit Leading Bit	63 Bit Mantissa
------------	-----------------	------------------------	-----------------

There is no implied bit. Intel takes 3 contiguous 32 bit words (12 bytes) to hold this.

### **LONG DOUBLE [SPARC Double Extended Precision]{16 Bytes}:**

This is the SPARC instruction set representation within the CPU where the actual arithmetic operations are done.

1 Sign Bit	15 Bit Exponent	1 Explicit Leading Bit	111 Bit Mantissa
------------	-----------------	------------------------	------------------

There is no implied bit. The SPARC architecture takes 4 contiguous 32 bit words (16 bytes) to hold this.

The long double is part of the IEEE-754 extended format definition. The extended definition allows for 80, 96 and 128 bit representations. The Intel IA-64 architecture only implements the 80 bit format. The SPARC architecture uses the 128 bit version.

### **FLOATING POINT NUMBERS WITHIN EXCEL**

Floating point numbers in Excel are stored in memory as double. Any internal computer operation on a double, using an Intel CPU, expands the double to a set of 3 contiguous 32 bit registers in the CPU. The format corresponds to the long double with a 15 exponent

---

<sup>2</sup> The interpretation here is first the IEEE-754 name, [ ] the Intel chip name, () the Excel VBA type (DIM) name and () the number of bytes required for storage. Note that FLOAT only occurs in a VBA function. Its use to provide a value for a worksheet cell may lead to inaccuracies and faults. The LONG DOUBLE is not supported in VBA functions or in cell equations. The LONG integer is different from the LONG DOUBLE. Worksheet functions only return double, Boolean, string or variant types. Integers in cells are actually double floating point values

and a 64 bit mantissa. Zero bits are inserted in the transfer. On RISC and HP computers without the long double capability, there is a transfer to two 32 bit registers. Floating point operations of division, square root, conversions to integer forms and conversions to other variant forms are done in software.

In the case of the Intel IA architecture, long double computations are done using 3 32 bit registers, and the outputs are placed in another 3-32 bit register set. The mantissa of 64 bits includes the addition of the implied bit to the first bit of the 64 bit set and then the following 63 bits are filled with the double mantissa (52 bits) and a fill in of zero bits. Following the computation on Intel CPUs as an extended double, the extended double is converted and rounded to a double for storage in memory. Information on how this is all done is in Cornea-Hasegan and Navin, 1999.

Changes to the other bits as a result of addition, subtraction, multiplication or division in the CPU can result in an un-normalized or de-normalized arrangement of bits, which do not correspond to the IEEE-754 standard. An automatic internal process of normalization occurs, which restores the bit patterns to what is referred to as the normalized state. Here the mantissa bits are moved left one at a time and the exponent bits are re-set until the left most bit is a one. Then one more shift is made, transforming this one-bit of information to the implied bit. Zero bits are added on the right to fill out the mantissa. In some cases a normal form cannot be obtained (primarily from invalid operations, denormal operands, divide-by-zero, overflow and underflow). Table 3-5 gives the common Excel return codes when these occur.

At the small-value-limit of a double, when the exponent bits are all zero, the exponent is treated as being fixed at  $-1022$ . The mantissa is then in the un-normal state, so that when all the bits of the mantissa are zero, the value zero is assigned to the number. (Pearson 1998). There are also sub-normal and NaN (not-a-number) bit patterns that can occur.

“Every decimal integer can be exactly represented by a binary integer; however, this is not true for fractional numbers. In fact, every number that is irrational in base 10 will also be irrational in any system with a base smaller than 10.

“For binary, in particular, only fractional numbers that can be represented in the form  $p/q$ , where  $q$  is an integer power of 2, can be expressed exactly, with a finite number of bits.

“Even common decimal fractions, such as decimal 0.0001, cannot be represented exactly in binary. (0.0001 is a repeating binary fraction with a period of 104 bits).” (KBA 78113)

### **ARITHMETIC ON FLOATING POINT NUMBERS**

Errors that occur during computer arithmetic {IF} operations are:

Round off error. Results when addition, subtraction, multiplication or division occurs. Also occurs when the sequences involve interchanges between internal 80 bit registers and external 64 bit memory storage.

Overflow and underflow. Results when the sequence of instructions results in one of the intermediate values either exceeding  $fp_{max}$  or being less than  $fp_{min}$ . An error return does not always occur. Changing the associations will result in different results.

Quantizing error. Results when the decimal number cannot be exactly represented by the IEEE-754 binary representation.

Sub-normal, under-flow, over-flow and not-a-number: These are other bit patterns that can occur from computations or conversions. When they occur, Excel returns the NaN designation or a zero

{IF} numbers do not follow mathematical theory or match expectations of {IR}. However, in Excel, the computer treats a zero as a mathematical object, rendering all products as zero and all divisions either as a zero or as the error code “/0”, just like {IR} arithmetic.

The accuracy of Excel computations is limited by the computer arithmetic. The display of a maximum of 15 decimal digits allows for some of the errors from computer arithmetic to not be seen. For extended computations, the computer arithmetic error cannot be predicted. Therefore Excel is faulted, when it is expected to do {IR} arithmetic exactly with 15 correct decimal digits from extended arithmetic every time. To ensure accurate 15 or more decimal digit computations, one has to use the xnumbers add-in (see Note XA) or resort to other programs such as Mathematica.

KBAs 42980, 78113, 145889, 125056 and 214118 are some good sources of information on the {IF} problem. McCullough (1998) in his first article also discusses this problem. Knuth (1998) in his books on the art of computer programming presents the basic theoretical problems of accurately adding, subtraction, multiplying and dividing using floating point numbers as the {IF} object. Higham (1993) also finds that there is no universal way to correct for addition (and subtraction) rounding errors in long lists in floating point form.

### **THE PERCEPTION OF NUMBERS**

There is a conflict between the way we as humans view a floating point number and just what the number is. For example, a display of a number as 3E+30 is viewed as

3,000,000,000,000,000,000,000,000,000.

being the number 3, followed by 30 zeros and a decimal point. This would be an {IR} perception. The {IF} construct would be:

3000000000000000XXXXXXXXXXXXXXXXXX.

Where the X's represent totally unknown, unidentified digits. This of course would not be in the form of decimal numbers as shown, but in bits representing the first 16 numbers in a binary form.

Now suppose we do arithmetic (addition) on the set of 3 numbers, 3E+30, 3E+30 and 3E+30, we would expect:

9,000,000,000,000,000,000,000,000,000.

What we get is:

9000000000000000XXXXXXXXXXXXXXXXXX.

The last X decimal digits are now uncertain.

If instead the standard deviation function (STDEV) is applied to the 3 numbers, we get

534274778125631XXXXXXXX.

Instead of zero. This number just represents a one bit difference in the mantissa due to {IF} arithmetic. This is not a fault of Excel, but a demonstration of how {IF} arithmetic affects our perceptions on accuracy.

### **ZEROS, AND SMALL VALUES**

The smallest double precision number displayed in Excel is a threshold value of  $2.2250738585072E-308$  (rmin). Above this threshold all 15 decimal digits of the double precision number are supported (KBA 78113). Below this threshold, any value is displayed as zero, and  $0.000000000000000E+00$  or  $0.000000000000000E-01$  appears

The smallest double precision number that can be held in the 64-bit format is  $4.940656458412465E-324$  (fpmin). This number is represented by the 51 high order mantissa bits all being zero, the hidden bit is one and the single low bit being one. The hidden bit is a virtual bit, which is always incorporated in a conversion to a decimal digit display. The smallest increment that can be made to any floating-point number is fpmin. Zero is when this last bit is set to zero, and all 64 bits are zero. Zero is a positive zero, since the highest bit (0 in this case) signifies that the number is positive.

The smallest extended precision number that can be held in the 80-bit format is  $3.36210314311209350626E-4932$ . For the SPARC 128 bit format, the number is essentially the same, only there are 33 decimal digits instead of the 21 shown. This shows that computations such as minimizations or maximizations (i.e. maximum likelihood) that result in very small numbers, will correctly compute within the Intel ix86 or SPARC registers, but are converted in memory to zero. This can create very substantial errors and problems with some statistical computation methods. (McCullough and Vinod 2003 and 2004).

This number  $4.940656458412465E-324$  (fpmin), in double precision is represented by the 51 high order mantissa bits all being zero, the hidden bit is one and the single low bit being one. The hidden bit is a virtual bit, which is always incorporated in a conversion to a decimal digit display. The smallest increment that can be made to any floating-point number is fpmin. Zero is when this last bit is set to zero, and all 64 bits are zero. Zero is a positive zero, since the highest bit (0 in this case) signifies that the number is positive.

The threshold number  $2.225073858507E-308$  (rmin) represents the case where all the 11 exponent bits of the double precision word are zero and the 52 mantissa-bits are all ones. Only the 52 bits of the mantissa represent any value between this and the smallest number. Between this value and the smallest value, the number of significant digits of the decimal representation goes from 15 to near zero. Microsoft chose this for Excel, as the minimum, since any value below this has less than 15 significant decimal digits. Theoretically the threshold gives rise to both positive or negative zeros, which is of computer science interest. To simplify analysis, any value below the threshold number will be interpreted as a signless zero from the function. This threshold number is Knüsel's (1999) "rmin" value.

Any number below this value is converted to a zero value in memory.

## **THE IMPORTANCE OF CORRECTLY HANDLING AND COMPUTING SMALL NUMBERS**

The issue of being able to correctly represent small numbers in Excel is important in statistics. Much of the use of statistical methods has to do with minimizing the likelihood function (see Edwards 1992 for a good view). Maximizing the likelihood function is the basis of solving structural equations (Bollen 1989), which are very extensively used in social research, marketing, psychology, education, economics, etc. In many cases, maximizing the log-likelihood function involves differences in a 15 digit floating point number in the last 1 or 2 digits (see McCullough and Vinod 2003 and 2004) at or near to the  $r_{min}$  value. The fitted parameter values then are quite inaccurate, due to this small variability. If the number is below  $r_{min}$ , a zero value is put into the corresponding memory location, effectively blocking the attainment of any meaningful solutions.

Likelihood equations can be built using Excel cells. If Solver (see section 10) is used as a minimizer, the erratic behavior of Solver in calculating the likelihood value totally overwhelms the ability to accurately find a stable, correct maximum. If an add-in is used that can correctly perform a minimization, it too depends on correct gradient values, which in turn are inherently very small values.

This involves essentially an argument that Excel needs to be able to store and display the IEEE-754 extended double floating point number, so it can be correctly reused in subsequent calculations.

This inability is an inherent Excel fault.

## **MAXIMUM VALUES**

The largest double precision number is  $1.797693134862315E+308$  (fpmax). Although Excel will only allow numbers less than  $1E+308$  to be entered, equation calculations will allow results up to  $1.797693134862315E+308$ . Beyond this, #NUM! is returned.

There is a maximum limit on the accuracy of floating point numbers. For the case of whole numbers (integers) and when the hidden bit and all 52 bits of the mantissa are 1 (53 bits), the number would be  $2^{54} - 1$  or 18,014,398,509,481,983. This number has 17 decimal digits. These are not full range digits, and the convention is that the maximum value has only 16.5 decimal digits. When this number is entered into a cell, the cell (after Enter) shows 18,014,398,509,481,900. Excel on entry only accepts the leftmost 15 digits. The maximum 18,014,398,509,481,983 value can then only be attained by calculation. When displayed, Excel will show the calculated number rounded as 18,014,398,509,482,000. Beyond this, the whole number (as bits) is only represented by the high order bits, and the low order bits are lost. The number 18,014,398,509,481,983 ( $1.8E+16$ ) will be referred to as wnmax.

## **INHERENT FLOATING POINT LIMITS TO ACCURACY**

To summarize then, the limits on results from calculations using floating-point numbers depend on whether fpmin, fpmax and wnmax are encountered in the calculation sequences. When data values are larger than wnmax, there is a distinct loss of information, and Excel Functions will tend to give results different from expected results. Functions that have given correct values will suddenly show incorrect values when input numbers are larger than wnmax, or when intermediate values are larger than wnmax. In

many cases, this error can be corrected for, but Microsoft overlooked this when it built the new Excel 2003 functions.

Visual Basic for Applications (vba) which is the basic background computing “machine” for Excel, does not recognize the  $r_{min}$  limit as a zero. Therefore one may find some errors when doing spreadsheet calculations with very small numbers.

Another frequent problem surfaces when data with a large magnitude is entered, and the data has small variations (on the order of  $10^{-15}$  times the data magnitude or the data exceeds  $w_{max}$  and the variations are below  $w_{max}$ ). Here the results of AVERAGE, VAR and STDEV and related functions may show definite errors. These errors come from the fact that {IR} arithmetic represents the true value, and the difference from {IF} arithmetic is considered an error. In this case the error may not be correctable.

### **SEQUENCES OF FLOATING POINT CALCULATIONS**

In {IF} arithmetic,  $f_{min}$  is considered fully accurate, even though its conversion to decimal representation appears to have little accuracy. Therefore  $f_{min}$  as an “accurate” input number is correct. However in a sequence of multiplications and divisions (in binary) intermediate values will retain accuracy as long as any intermediate terms do not exceed  $w_{max}$  or are less than  $r_{min}$ . If  $w_{max}$  is exceeded or the intermediate value is less than  $r_{min}$ , the accuracy of the results will be heavily dependent on the way the sequence was structured (This is the {IF} association problem).

If  $f_{min}$  or  $f_{max}$  occurs, a #NUM may or may not be returned, depending on how the sequence is programmed, and if all the intermediate results can be retained in an 80-bit floating point register.

## **DISPLAYS AND OUTPUT**

### **DISPLAY OF THE {IF} OBJECT**

Communications with the human user are done with the calculated number being viewed on the screen or being printed, in the form of decimals. The user cannot see the {IF} object directly. The numbers seen on the screen are characters, with the string property. There are internal algorithms that convert the {IF} object (a variant) to a string of characters {DISP}, which the viewer interprets as numbers. The {DISP} object then can be changed to be shown in different languages, fonts, sizes, colors, locations on the screen and into different representations of the {IF} object (such as percentages),

What is displayed may not always be the exact version of the {IF} object. The conversion is done by display software that converts the binary number into a decimal display. In this link, registry keys and related software are involved. In at least two instances, Excel faults were found to be errors in the registry, which required downloading of an Excel “update” to correct. There was no fault in the computations involving the “numbers”, it was in the memory “number” conversion-to-display process. In the first case it was about an internal limit on the number of digits of a  $p$  value, and in the last case it was about the number “63535” not being correctly displayed when it was the result of a computation.

When there is no fault in the display process, then the “number” is an exact conversion of the variant to a “display” according to the “format” criteria set for that cell. Rounding and all the other criteria are taken into account.

The display of the {IF} object has turned out to be a primary consideration in all the published articles on Excel.

Monahan (2004) states that, “students detest any material on computer arithmetic”. They are annoyed when the display is 3.99999, but fully accept a display of 4.0. He says that the commercial software packages have “artfully” covered up the computer arithmetic problems by the output display. Monahan’s comments just support the view that the display either supports or establishes preconceived notions on computer outputs.

### **EXCEL 2003 AND 2007 DEFAULT DISPLAYS<sup>3</sup>**

When Excel 2003 is started with a blank workbook, worksheet displays are shown in a default mode. The default mode is that “cell formats” are “General”, “Column widths” are “8.43” pixels, and the cell “font” is “Arial”, size “10”. These are the main “controls” that set in the number of digits displayed in a cell. Changing any of these changes the number of displayed digits. This is shown in tables 3-2, 3-3 and 3-4, where each table represents a different font size.

There are other controls, which in general (there are exceptions) do not have an effect on the number of digits shown in a cell.

**Table 3-2: Excel 2003 Default Display, {DISP} Objects, Arial Font 10 (default)**

<b>Input {IF}Object</b>	<b>Column Width, Minimum, 5.43 points</b>	<b>Column Width, Default, 8.43 points</b>	<b>Column Width, Wide, 15.43 points</b>
12345678912345	1E+13	1.23E+13	1.23457E+13
123456789123	1E+11	1.23E+11	1.23457E+11
12345678912	1E+10	1.23E+10	12345678912
1234567891	1E+09	1.23E+09	1234567891
123456789	1E+08	1.23E+08	123456789
12345678	1E+07	12345678	12345678
1234567	1E+06	1234567	1234567
123456	1E+05	123456	123456
12345	12345	12345	12345
1234	1234	1234	1234
123	123	123	123
12	12	12	12
1	1	1	1
0.123456789	0.123	0.123457	0.123456789
0.0123456789	0.012	0.012346	0.012345679
0.00123456789	0.001	0.001235	0.001234568
0.000123456789	1E-04	0.000123	0.000123457
0.0000123456789	1E-05	1.23E-05	1.23457E-05

<sup>3</sup> Excel 2007 retains the basic Excel 2003 characteristics, except appearances. Findings on Excel 2003 also apply to Excel 2007.

Table 3-2 is important, since it gives the Excel **Default** display.

The basic default display shown in column 3 has turned out to be the sole basis for criticisms in the literature on the accuracies of computed numbers in Excel in relation to the number of digits displayed. This dependence on default display also has occurred in articles on the errors in other software packages. For example, the tests on JMP 4.03 by Altman (2002) were found to be based on the JMP default display, not on the computational algorithms (Hilbe 2002). The paper by Creighton and Ding (2002) gives a more accurate assessment of JMP, and gives values different from that reported by Altman (2002). The differences between LRE values found in all of McCullough and Wilson's papers on Excel and the LRE values reported in this paper are mainly due to McCullough's use of the Excel default display as the Excel output.

The other columns show that even minor changes in column width can change the number of displayed digits.

**Table 3-3: Excel 2003 Default Display, {DISP} Objects, Arial Font Size 12**

Input {IF}Object	Column Width, Minimum, 5.43 pixels	Column Width, Default, 8.43 pixels	Column Width, Wide, 15.43 pixels
12345678912345	#####	1E+13	1.23457E+13
123456789123	#####	1E+11	1.23457E+11
12345678912	#####	1E+10	12345678912
1234567891	#####	1E+09	1234567891
123456789	#####	1E+08	123456789
12345678	#####	1E+07	12345678
1234567	#####	1E+06	1234567
123456	#####	123456	123456
12345	#####	12345	12345
1234	1234	1234	1234
123	123	123	123
12	12	12	12
1	1	1	1
0.123456789	0.12	0.1235	0.123456789
0.0123456789	0.01	0.0123	0.012345679
0.00123456789	0	0.0012	0.001234568
0.000123456789	0	0.0001	0.000123457
0.0000123456789	0	1E-05	1.23457E-05

**Table 3-4: Excel 2003 Default Display, {DISP} Objects, Arial Font Size 8**

Input {IF}Object	Column Width, Minimum, 5.43 pixels	Column Width, Default, 8.43 pixels	Column Width, Wide, 15.43 pixels
12345678912345	1E+13	1.235E+13	1.23457E+13
123456789123	1E+11	1.235E+11	1.23457E+11
12345678912	1E+10	1.235E+10	12345678912
1234567891	1E+09	1.235E+09	1234567891
123456789	1E+08	123456789	123456789
12345678	1E+07	12345678	12345678
1234567	1E+06	1234567	1234567
123456	123456	123456	123456
12345	12345	12345	12345
1234	1234	1234	1234
123	123	123	123
12	12	12	12
1	1	1	1
0.123456789	0.1235	0.1234568	0.123456789
0.0123456789	0.0123	0.0123457	0.012345679
0.00123456789	0.0012	0.0012346	0.001234568
0.000123456789	0.0001	0.0001235	0.000123457
0.0000123456789	1E-05	1.235E-05	1.23457E-05

When the user clicks on “Format” and “Cells”, a menu comes up in which the user can select one of many ways to display the {IF} object instead of the default display. When Numbers are selected, the number of decimals to be displayed is also entered. For a floating point {IF} object up to 15 decimal digits can be selected. Excel automatically inserts zeros when the total number of digits to be displayed exceeds 15.

This is identical to the FORTRAN process of a numbered FORMAT () statement, where such codes as F8.2, I5 etc. designate how the {IF} numbers in the PRINT line are to be printed. (Note: Excel has a similar capability, see Note K.)

Consequently how the {IF} object is displayed is a factor in deciding on the accuracy of the computation. If there are true decimal zeros in the string, they cannot be distinguished from display added zeros, if the zeros are trailing to non-zero digits. Display methods such as rounding, choices of format and number of digits to the right of a decimal point, all contribute to the perception of a reader on the accuracy of the displayed number.

## **NON-NUMERIC ERROR MESSAGE RETURNS**

In {IF}, errors occur and codes are returned rather than numbers when these occur. The common error messages that appear are given in table 3-5.

**Table 3-5: Excel Common Error Messages**

<b>Symbol</b>	<b>Description</b>
#DIV/0!	The function/equation is trying to divide by zero. This also occurs when the referenced cell is empty. This is the division by zero trap expressed by an output symbol.
#NAME?	The function name is not recognized, either within Excel or in the VBA routines that are part of the workbook. Also occurs when unmated quotes around text occur.
#N/A	The function inputs refer to a cell that uses the NA functions to signal the fact that the data is not available. Occurs when the input accepts a range of values for two inputs and they do not have the same number of data points.
#NULL!	The formula uses an intersection of two ranges that don't intersect.
#NUM!	There is an internal function problem with obtaining an output number. Also occurs when input parameter values are not within the specified parameter ranges for that function variable. (Function input value checking). Also occurs with inverse functions when the number of internal iterations exceed a preset number. See KBA 827743 for a valid situation.
#REF!	The function has an input cell reference that is not valid (cell deleted?).
#VALUE!	There is a mismatch somewhere between the function required input parameter type and the parameter type of the referenced cell (or value). Frequently occurs when the actual input is non-numeric or the referenced cell contains non-numeric data. Also returned when a VBA macro uses Excel functions (i.e. Application.WorksheetFunction.XXXX()) and the Tools-option-calculation is manual and F9 not pushed.). Also returns from any macro with an internal error. Also returned when the function is not in any associated VBA modules (can't find the function). The Invalid Operation trap (NaN, not a number) also gives this symbol.
#	Outputs that begin with the pound sign. If the cell is filled with #, it indicates that the cell is not wide enough to display all the data. Easily fixed by increasing column width.

The IEEE-754 standard defines five error traps (flags). They are

1. Invalid operation (not a number, NaN),
2. Overflow ( $\pm \infty$ ), sign depending on rounding direction
3. Division by zero ( $\pm \infty$ ), sign from the numerator
4. Underflow, some tiny number
5. Inexact result (rounded, overflowed or under flowed).

Excel does not directly output these traps. As a result, Excel cannot be taken as an “Algebraically Complete” system. (Kahan 2004). “In the absence of round off and over/underflow, evaluation of an algebraic expression that differ because the customary commutative, distribution, associative and cancellation laws have been applied can yield at most two values and, if two, one must be NaN (Not a Number)”. Here zero and infinity are considered numbers. (Kahan 2004). Kahan however goes on to say that when a cell

number is not finite, subsequent algebraic cancellations (or equality tests) become invalid and invalidate any subsequent computations and decisions made about the results.

“Perhaps no traces will be left to arouse suspicions that plausible final results are actually quite wrong. Therefore a program must be able to detect that non-finite values have been created, in case it has to take steps necessary to compensate for them.”

The occurrence of these error messages may or may not be a frequent problem to a user, but it just prevents the user from obtaining a desired value. Frequent occurrences of #NUM! when the inputs are valid, indicates poor reliability. #VALUE errors can be corrected by fixing inputs, but a #NUM! result usually can't be fixed.

McCullough (2000) argues that Excel should return the “ns” symbol (for “no solution”) when a result cannot be calculated. In the article, he was specifically referring to tests on fitting non-linear equations to data. In this case, the inability to find a fit would return “ns” instead of a number. In Microsoft's routines, the “#NUM!” symbol indicates a “no solution”, either from an incorrect input, or from some internal function test, such as the exceeding of a number-of-iterations limit. Solver, which is used for non-linear equation fits, is not a Microsoft product (see section 8), and returns other signs when “no solution” occurs. In neither of these cases, does the function/routine return a no-solution when the value/values are “inaccurate”. McCullough (2000) would prefer a “no-solution” return rather than an “inaccurate” value.

There is an inherent logical problem here with “no-solution”, and the problem is “how is no-solution determined?” when the correct solution is not known prior to the computation. All that one has is either a number or a flag. In the case of a number, there are no grounds to say the number is incorrect, unless some prior bounds are known, which can be used to detect a “no-solution”. This does not provide the general solution to the “no-solution” problem.

The “no-solution” return then requires some “oracle” in the algorithm that internally can decide when a “no-solution” occurs. The only tool we have is a count on the number of iterations, when the algorithm has internal “trial and error” loops. LINEST which is a direct computation does not have any internal loops, but is a direct matrix reduction. If there are no #NUM occurrences, there is no way to deduce that a “no solution” occurred.