Computer Arithmetic

CEE3804: Computer Applications for Civil and Environmental Engineers

Learning Objectives

- Define: bit, byte, machine epsilon, exponent, significand, mantissa, overflow, underflow,
- Contrast integer vs floating point storage.
- Describe how range and precision varies between single and double precision.

How computers store numbers:

- Computer arithmetic is not the same as pencil and paper arithmetic or math class arithmetic.

- Hand calculations usually short. Small errors negligible. Computer calculations longer, may accumulate errors over millions of steps to catastrophic results. Software itself can be buggy.

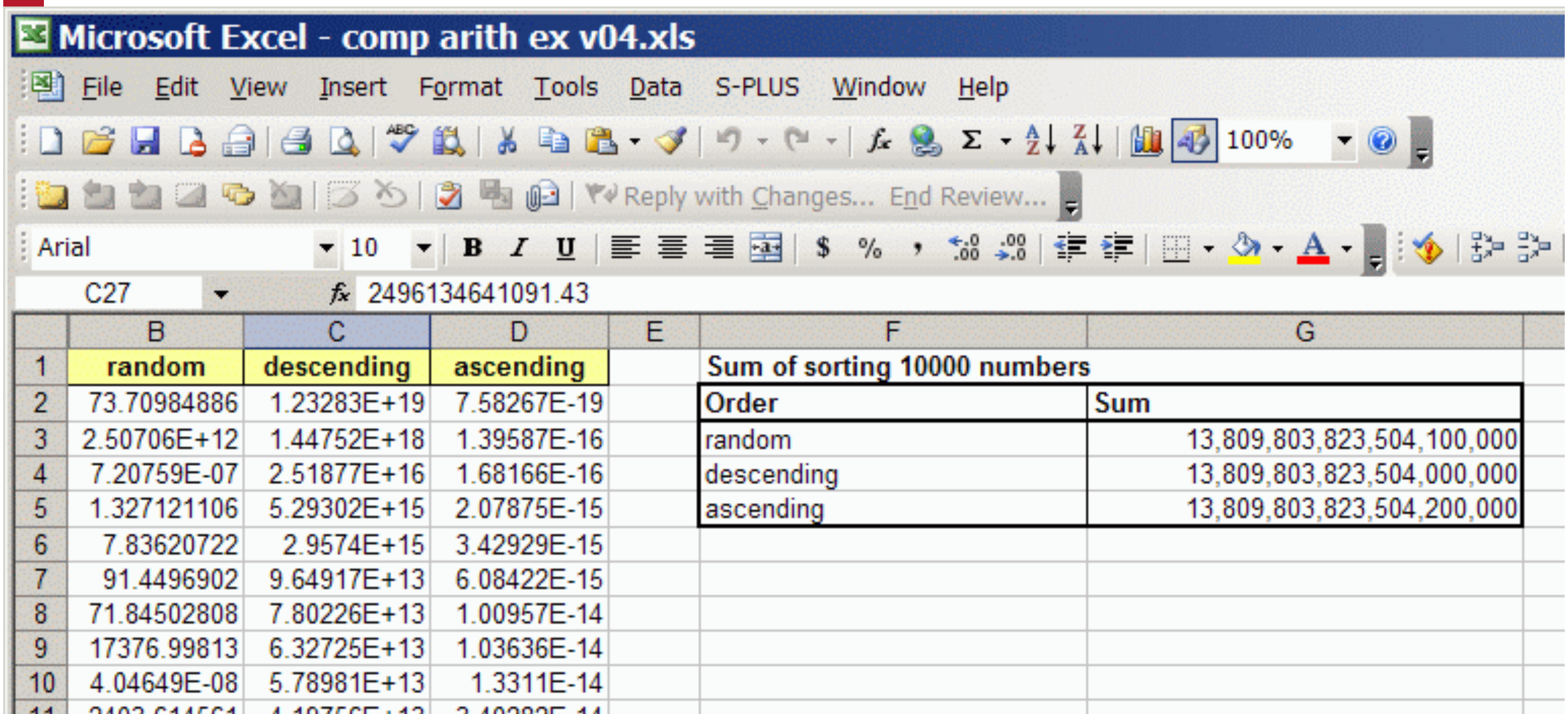
Errors in scientific computing

- A. machine hardware malfunctions
  - Very rare, but possible. Recall Pentium floating point error.
- B. software errors
  - More common than you might think.
  - see calc.exe
    Windows 3.1 calculator. Subtract 3.11 - 3.1 = 0.00. (Note the answer is calculated correctly but displayed incorrectly. You can check this by multiplying the answer above 0.00 * 100 = 1.)
  - See http://www.wired.com/news/technology/bugs/0,2924,69355,00.html

Errors, continued

- C. blunders - programming the wrong formula
  - Depending on the QA/QC implemented, can be very common.
  - These errors can arise from typos or other outright errors. experimental error - data acquired by machine with limited precision
- D. Truncation error
  - A floating point number often cannot be represented exactly by the computer. Only a fixed storage length is available. Often a portion of the number must be truncated or rounded.
  - Example: sums of a series of numbers vary depending on the order in which they are added.

Sorting Error Example



	B	C	D
1	random	descending	ascending
2	73.70984886	1.23283E+19	7.58267E-19
3	2.50706E+12	1.44752E+18	1.39587E-16
4	7.20759E-07	2.51877E+16	1.68166E-16
5	1.327121106	5.29302E+15	2.07875E-15
6	7.83620722	2.9574E+15	3.42929E-15
7	91.4496902	9.64917E+13	6.08422E-15
8	71.84502808	7.80226E+13	1.00957E-14
9	17376.99813	6.32725E+13	1.03636E-14
10	4.04649E-08	5.78981E+13	1.3311E-14

Sum of sorting 10000 numbers

Order	Sum
random	13,809,803,823,504,100,000
descending	13,809,803,823,504,000,000
ascending	13,809,803,823,504,200,000

Truncation Error Example

Microsoft Excel - comp arith ex v04.xls

File Edit View Insert Format Tools Data S-PLUS Window Help

	A	B	C	D	E	F	G
1	**Truncation example**						
2	Calculate variance (s^2) of 3 numbers						
3		**Example 1**	**Example 2**	**Example 3**	**Difference**	**Col C squared**	**Col D squared**
4		0	9,999,999	9,999,999,999,999	0	99999980000001	99999999999980000000000000
5		1	10,000,000	10,000,000,000,000	1	100000000000000	100000000000000000000000000
6		2	10,000,001	10,000,000,000,001	2	100000020000001	100000000000020000000000000
7							
8	variance	1	1	17179869184	1		
9							
10	All the columns should have the same variance.						
11							
12	Variance is typically calculated as						
13	var= summation (x_j^2)-N*xbar^2						
14	Recall that double precision stores 15-16 digits. For column D, when the terms are squared,						
15	the terms loose the last digit which is where the variability should appear						
16							

Errors, continued

- E. numerical or rounding error
  - 1. ill conditioning or sensitivity of problem
    - For example, finding the intersection of 2 nearly parallel lines.
  - 2. stability of algorithm
    - Can also use inappropriate algorithm. Example: Taylor series expansion to evaluate exp(x).

$$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!} + \ldots$$

    - Works for positive numbers but fails for large magnitude negative numbers because of excessive cancellation errors.

Rounding Error, continued

- If use this algorithm to solve for exp(-25), the following iterations results using single precision on an IBM PC. The solution converges to 142.3876.

- The correct answer is

  exp(-25) = 1.38879x10-11

Rounding Error, Example

Iteration	Value	Iteration	Value	Iteration	Value
1	-24	31	-1.165549E+09	61	131.7048
2	288.5	32	8.946474E+08	62	146.646
3	-2315.667	33	-6.661073E+08	63	140.7169
4	13960.38	34	4.815065E+08	64	143.033
5	-67419.84	35	-3.382176E+08	65	142.1422
6	271664.4	36	2.310352E+08	66	142.4796
7	-939350.8	37	-1.535951E+08	67	142.3537
8	2845072	38	9.945117E+07	68	142.4
9	-7667213	39	-6.275797E+07	69	142.3832
10	1.86135E+07	40	3.862274E+07	70	142.3892
11	-4.111539E+07	41	-2.319476E+07	71	142.3871
12	8.331979E+07	42	1.360137E+07	72	142.3878
13	-1.559786E+08	43	-7791729	73	142.3876
14	2.7134E+08	44	4363444	74	142.3877
15	-4.408577E+08	45	-2389430	75	142.3876
16	6.719512E+08	46	1280610	76	142.3876
17	-9.645325E+08	47	-671538.9	77	142.3876
18	1.308361E+09	48	345205.3	78	142.3876
19	-1.682288E+09	49	-173541.7	79	142.3876
20	2.056024E+09	50	85831.8		
21	-2.394348E+09	51	-41312.08		
22	2.662893E+09	52	19814.79		
23	-2.834108E+09	53	-9018.639		
24	2.891934E+09	54	4330.171		
25	-2.834108E+09	55	-1737.469		
26	2.671702E+09	56	971.2986		
27	-2.42627E+09	57	-216.7576		
28	2.125491E+09	58	295.3356		
29	-1.798441E+09	59	78.34695		
30	1.471502E+09	60	168.7589		

Significant Figures

- The significant digits of a number are those that can be used with confidence. They correspond to the certain digits plus one estimated digit.
- For example, a metric ruler marked to millimeters would have significant digits to the nearest tenth of a millimeter.

Accuracy

- Accuracy refers to how closely a computed or measured value corresponds to the true value. Since the true value is almost always unknown, accuracy is rarely known. Sometimes bounds can be placed on how accurate (or inaccurate) a calculation is.

Precision

- **Precision refers to how closely individual computed or measured values agree with each other.**

Absolute vs Relative Error

- **True value = approximation + absolute error**
- **absolute error = |true value - approximation|**

$$relative\ error = \left| \frac{true\ value - approximation}{true\ value} \right|$$

Absolute vs Relative Error, cont.

- In practice, don' t know true value and use best available estimate
- absolute error = current estimate - previous estimate

$$relative\ error = \left| \frac{current\ estimate - previous\ estimate}{current\ estimate} \right|$$

Numerical Data Types: Integers

- Most computers (but not all) use base 2.

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
128	64	32	16	8	4	2	1

- Thus 101 base 2 = 5
  1100 base 2 = 12
- 1 bit = binary storage location with only 2 possible states: 0/1 or +/-
- 1 byte = 8 bits

Numerical Data Types: Integers

- Simple way to convert from binary to decimal
- Find the equivalent number in base 10 for 1100 in base 2
- Each binary corresponds to a value multiplied by two and raise to the power n
- $(1 * 2^3) + (1 * 2^2) + (0 * 2^1) + (0 * 2^0)=12$
- Find the largest number that can be stored in one byte (8 bits).

Integers, continued

- Simply stored as base 2 number with 1 bit allocated to sign

+/-	2^6	2^5	2^4	2^3	2^2	2^1	2^0

+/-	1	1	1	1	1	1	1	=+/- 127

Size	Range	
1 bytes	-127 .. 127	
2 bytes	-32,768 .. 32,767	integer
4 bytes	-2,147,483,648 .. 2,147,483,647	long

Numeric Data Types: Floating point (reals)

- Stored as approximation only

Size	Range	Significant Digits
4 bytes	$1.18x10^{-38}$ .. $3.4x10^{38}$	7 - 8 (single)
8 bytes	$2.2x10^{-308}$ .. $1.7x10^{308}$	15 - 16 (double)
10 bytes	$3.4x10^{-4932}$ ..$1.1x10^{4932}$	19 - 20 (extended)

These particular examples are commonly implemented in the hardware and are processed relatively quickly. However, any size and therefore range, is possible by implementing the storage in software only.

Floating Point

- Floating point number is stored as 3 parts:
  - 1) sign (+ or -)
  - 2) exponent
  - 3) significand or mantissa
- A represented number conceptually has the value

  $+/-$ significand x 2^{exponent},

  where $0 <= \text{mantissa} < 2$
- (In practice, mantissa has single bit to the left of the binary decimal point, exponent is biased to save space for sign)

Floating points, continued

- **Example binary storage for a 4 byte number (4 bytes = 32 bits)**

1 bit (sign)	8 bit (exponent)	23 bit (mantissa or significand)

Three key computer values

- 1) UFL underflow
  The smallest nonzero number (power of 2) that can be stored. (Some applications set FP < UFL to 0, others stop with error.)

- 2) OFL overflow
  The largest number (power of 2) that can be stored. (Most applications consider FP > OFL to be error.)

Machine Epsilon

- 3) e_m machine epsilon
  The smallest number (power of 2) that when added to 1 is greater than 1.

$$1.0 + e_m > 1.0$$

For FP < e_m,

$$1.0 + FP = 1.0$$

$$1.0E0 + 1.0E\text{-}8 = 1.00000001 \implies 1.0E0$$

Numeric parameters, continued

- In general, OFL and UFL are determined by the number of bits used to store the exponent.
- e_m is determined by the number of bits used to store the significand.

e_m in Excel

	epsilon	1+epsilon	test	
1	1.00E-08	1.0000000100000000	different than 1	
1	1.00E-09	1.0000000010000000	different than 1	
1	1.00E-10	1.0000000001000000	different than 1	
1	1.00E-11	1.0000000000100000	different than 1	
1	1.00E-12	1.0000000000010000	different than 1	
1	1.00E-13	1.0000000000001000	different than 1	
1	1.00E-14	1.0000000000000100	different than 1	
1	1.00E-15	1.0000000000000000	equal to 1	
1	1.00E-16	1.0000000000000000	equal to 1	

Excel example: machine epsilon

power of 2	-47
1+2^power = 1 ?	false

power of 2	-48
1+2^power = 1 ?	true

Machine epsilon: Importance

- Determines relative accuracy of computer arithmetic. E.g. x,y positive FP numbers, x > y, sum written as

$$x + y = x\,(1 + y/x)$$

- Unless y/x > em, the FP sum of x and y will be x.

e_m importance, continued

- Note all numbers cannot be represented exactly in a given base. e.g. 1/3 cannot be written out exactly as a base 10 FP number. 0.3 cannot be written out exactly as a base 2 FP number.

- The error in reading in a decimal number can be as great as e_m.

- $x_{stored} = x(1 + dx)$ or $x_{stored} - x = dx$

  $|dx| <= e_m$

Example Values

- On an IBM PC
  - Single precision
    - UFL 2^-126 = 1.18E-38
    - OFL 2^128 = 3.40E+38
    - e_m 2^-23 = 1.19E-07
  - Double Precision
    - UFL 2.23D-308
    - OFL 1.79D+308
    - e_m 2^-52 = 2.22D-16
- On Sharp EL-506A calculator (based on display)
    - UFL 2^-328 = 1.83E-99
    - OFL 2^332 = 8.75E99
    - em 2^-30 = 9.31E-10

Implications of Floating Point Storage

- Only finite many floating point numbers, about 2^31 in single precision.
- There is largest floating point number - OVL
- There is smallest floating point number - UFL
- The floating point numbers between 0 and OFL are not evenly distributed. In single precision, there are 2^22 floating point numbers between each power of 2.

Example:

- 2^22 numbers between 2^-126 and 2^-125
  (1.17E-38 and 2.35E-38)

- 2^22 numbers between 2^125 and 2^126
  (4.25E37 and 8.50E37)

- Floating point numbers are concentrated near 0.

Implications, continued

- Arithmetic operations on floating point numbers cannot always be represented exactly, and must be either truncated or rounded to the nearest floating point number.
- e_m is smallest floating point number such that
  $1.0 + em > 1.0$
- e_m represents the relative accuracy of computer arithmetic.

Implications, continued

- OFL and UFL are determined mostly by the number of bits in the exponent. em is determined mostly by the number of bits in the significand (mantissa). Measure different parts of the floating point representation

- $0 < \mathrm{UFL} < \mathrm{em} < \mathrm{OFL}$