1.2 Round-off Errors and Computer Arithmetic

- In a computer model, a memory storage unit – **word** is used to store a number.
- A **word** has only a finite number of bits.
- These facts imply:
  1. Only a small set of real numbers (rational numbers) can be accurately represented on computers.
  2. (Rounding) errors are inevitable when computer memory is used to represent real, infinite precision numbers.
  3. Small rounding errors can be amplified with careless treatment.

So, do not be surprised that $(9.4)_{10} = (1001.\overline{0110})_2$ can not be represented exactly on computers.

IEEE floating point numbers

- Binary number: $(\dots b_3 b_2 b_1 b_0 . b_{-1} b_{-2} b_{-3} \dots)_2$
- Binary to decimal:
  $(\dots b_3 b_2 b_1 b_0 . b_{-1} b_{-2} b_{-3} \dots)_2 =$
  $(\dots b_3 2^3 + b^2\, 2^2 + b_1 2^1 + b_0 2^0 + b_{-1} 2^{-1} + b_{-2} 2^{-2} + b_{-3} 2^{-3} \dots)_{10}$
- Double precision (long real) format
  - Example: “double” in C
- A 64-bit (binary digit) representation
  - 1 sign bit (s), 11 exponent bits – characteristic (c), 52 binary fraction bits – mantissa (f)

x	xxxxxxxxxxx	xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
s	c	f

Represented number (Normalized IEEE floating point number):

$$(-1)^s 2^{c-1023}(1+f)$$

1023 is called exponent bias

$$0 \leq characteristic\ (c) \leq 2^{11} - 1 = 2047$$

- Smallest normalized positive number on machine has $s = 0, c = 1, f = 0$: $2^{-1022} \cdot (1 + 0) \approx 0.22251 \times 10^{-307}$
- Largest normalized positive number on machine has $s = 0, c = 2046, f = 1 - 2^{-52}$: $2^{1023} \cdot (1 + 1 - 2^{-52}) \approx 0.17977 \times 10^{309}$
- **Underflow**: $numbers\ < 2^{-1022} \cdot (1 + 0)$
- **Overflow**: $numbers > 2^{1023} \cdot (2 - 2^{-52})$
- **Machine epsilon** $(\epsilon_{mach}) = 2^{-52}$: this is the difference between 1 and the smallest machine floating point number greater than 1.

- Positive zero: $s = 0, c = 0, f = 0$.
- Negative zero: $s = 1, c = 0, f = 0$.
- Inf: $s = 0, c = 2047, f = 0$
- NaN: $s = 0, c = 2047, f \neq 0$

Decimal machine numbers

- Normalized k-digit *decimal machine* numbers:
  $\pm 0.d_1 d_2 \ldots d_k \times 10^n, \qquad 1 \leq d_1 \leq 9, 0 \leq d_i \leq 9$
- Any positive number within the numerical range of machine can be written:
  $$y = 0.d_1 d_2 \ldots d_k d_{k+1} d_{k+2} \ldots \times 10^n$$

Chopping and Rounding Arithmetic:

Step 1: represent a positive number y by
$0.d_1 d_2 \ldots d_k d_{k+1} d_{k+2} \ldots \times 10^n$

Step 2:

– Chopping: chop off after k digits:
$$fl(y) = 0.d_1 d_2 \ldots d_k \times 10^n$$

– Rounding: add $(5 \times 10^{-(k+1)}) \times 10^n$ to y, then chopping

a) If $d_{k+1} \geq 5$, add 1 to d_k to get $fl(y)$

b) If $d_{k+1} < 5$, simply do chopping

Errors and significant digits

Definition

If p^* is an approximation to p, the *absolute error* is $|p - p^*|$, and the *relative error* is $|p - p^*|/|p|$, provided that $p \neq 0$.

Definition

The number p^* is said to approximate p to t *significant digits* (or figures) if t is the largest nonnegative integer for which

$$\frac{|p - p^*|}{|p|} \leq 5 \times 10^{-t}.$$

Finite-digit arithmetic

- Machine addition, subtraction, multiplication, and division:

$$x \oplus y = fl(fl(x) + fl(y)), \quad x \otimes y = fl(fl(x) \times fl(y))$$
$$x \ominus y = fl(fl(x) - fl(y)), \quad x \oslash y = fl(fl(x) \div fl(y))$$

- "Round input, perform exact arithmetic, round the result"

- **Catastrophic events**
  a) Subtracting nearly equal numbers – this leads to fewer significant digits.
  b) Dividing by a number with small magnitude (or multiplying by a number with large magnitude).

Avoiding loss of accuracy by reformulating calculations

Quadratic formula to find roots of $ax^2 + bx + c = 0, where\ a \ \neq 0$.

1. $x_1 = \frac{-b+\sqrt{b^2-4ac}}{2a}$
2. $x_2 = \frac{-b-\sqrt{b^2-4ac}}{2a}$

Key: Magnitudes of $b\ and\ \sqrt{b^2 - 4ac}$ decide whether we need to reformulate the formula.