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.

- Round-off error: error that is produced when a computer is used to perform real number calculations.

Binary numbers and decimal numbers

- Binary number system:

A method of representing numbers that has 2 as its base and uses only the digits 0 and 1. Each successive digit represents a power of 2.

$$(\dots b_3 b_2 b_1 b_0 . b_{-1} b_{-2} b_{-3} \dots)_2$$

where $0 \leq b_i \leq 1$, for each $i = \cdots 2,1,0,-1,-2$

- Binary to decimal:

$$(\dots b_3 b_2 b_1 b_0 . b_{-1} b_{-2} b_{-3} \dots)_2$$
$$= \begin{array}{l} (\dots b_3 2^3 + b_2\ 2^2 + b_1 2^1 + b_0 2^0 \\ + b_{-1} 2^{-1} + b_{-2} 2^{-2} + b_{-3} 2^{-3} \dots)_{10} \end{array}$$

Example 1: Convert the binary number $(101.11)_2$ to decimal number

Binary machine numbers

- IEEE (Institute for Electrical and Electronic Engineers)
  - Standards for binary and decimal floating point numbers
- For example, a “double” type real number uses 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

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

This 64-bit binary machine number gives a decimal floating-point number (Normalized IEEE floating point number):

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

where 1023 is called exponent bias.

- 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** $(\epsilon_{mach}) = 2^{-52} \approx 2.2204 \times 10^{-16}$: this is the difference between 1 and the smallest machine floating point number greater than 1. (roughly 16 decimal digits of precision)

- Positive zero: $s = 0, c = 0, f = 0$.
- Negative zero: $s = 1, c = 0, f = 0$.
- Inf: $s = 0, c = 2047, f = 0$
- NaN: $s = 0, c = 2047, f \neq 0$
- Machine epsilon $\epsilon_{mach} = 2^{-52}$.
  - Difference between 1 and the smallest floating point number greater than 1.

Example 2.

a) Convert the following binary machine number $(P)_2$ to decimal number.

$(P)_2 =$	0	10000000011	$10111000000000 \ldots 0$

b) What's the next smallest/largest machine number of $(P)_2$?

Decimal machine numbers

- Normalized decimal floating-point form:

$$\pm 0.d_1 d_2 d_3 \ldots d_k \times 10^n$$

where $1 \leq d_1 \leq 9$ and $0 \leq d_i \leq 9$, for each $i = 2 \ldots k$.

A. Chopping arithmetic:

1. Represent a positive number y as $0.d_1 d_2 d_3 \ldots d_k d_{k+1} d_{k+2} \ldots \times 10^n$
2. Chop off digits $d_{k+1} d_{k+2} \ldots$. This gives:

$$fl(y) = 0.d_1 d_2 d_3 \ldots d_k \times 10^n$$

B. Rounding arithmetic:

1. If $d_{k+1} \geq 5$, add 1 to d_k (round up), and then chop
2. If $d_{k+1} < 5$, do nothing (round down), and then chop

Remark: $fl(y)$ represents normalized decimal machine number.

Example 3. Compute 5-digit (a) chopping and (b) rounding values of $\pi = 3.14159265 \ldots$

- Definition. Suppose p^* is an approximation to p. The **actual error** is $p - p^*$. The **absolute error** is $|p - p^*|$. The **relative error** is $\frac{|p-p^*|}{|p|}$, provided that $p \neq 0$.
  - Remark. Relative error takes into consideration the size of value.
- Definition. The number p^* is said to approximate p to t **significant digits** if t is the largest nonnegative integer for which $\frac{|p-p^*|}{|p|} \leq 5\times10^{-t}$.

Example 4. Find absolute and relative errors, number of significant digits for:

(a) $p = 0.3000 \times 10^1$ and $p^* = 0.3100 \times 10^1$

(b) $p = 0.3000 \times 10^{-3}$ and $p^* = 0.3100 \times 10^{-3}$.

Example 5. Find the largest interval in which p^* approximates $\sqrt{2}$ with relative error at most 10^{-3}.

Finite-Digit arithmetic

- Arithmetic operation in a computer is not exact.
- Let machine addition, subtraction, multiplication and division be $\oplus, \ominus, \otimes, \oslash$.

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

Remark: A machine operation preform exact arithmetic on the floating-point representations of x and y, then convert the exact result to its finite-digit floating-point representation.

Example 6. $x = \frac{5}{7} = 0.714285\ldots, y = \frac{1}{3} = 0.333333\ldots$. Use 5-digit chopping arithmetic to compute $x \oplus y$. Find its absolute error and relative error, and number of significant digits.

Example 7. $x = \frac{5}{7}, u = 0.714251$. Use 5-digit chopping arithmetic to compute $x \ominus u$. Find its absolute error and relative error, and number of significant digits.

Calculations resulting in loss of accuracy

1. Subtracting nearly equal numbers gives fewer significant digits.
2. Dividing by a number with small magnitude or multiplying by a number with large magnitude will enlarge the error.

Example 8. Suppose z is approximated by $z + \delta$. where error δ is introduced by previous calculation. Let $\varepsilon = 10^{-n}, n > 0$. Estimate the absolute error of $z \oslash \varepsilon$.

Technique to reduce round-off error

- Reformulate the calculation.

Example 9. Compute the most accurate approximation to roots of $x^2 + 62.10x + 1 = 0$ with 4-digit rounding arithmetic.

- Nested arithmetic
  - Purpose is to reduce number of calculations.

Example 10. evaluate $f(x) = x^3 - 6.1x^2 + 3.2x + 1.5$ at $x = 4.71$ using 3-digit chopping arithmetic.