Ch 01-2: Errors

Dr. O. Phillips Agboola

Round-off Errors and Computer Arithmetic

- The arithmetic performed by a calculator or computer is different from the arithmetic in algebra and calculus courses.
- 2+2 = 4, 4·8 = 32, and $(\sqrt{3})^2 = 3$. *However, with computer arithmetic we* expect exact results for 2+2 = 4 and 4 · 8 = 32, but we will not have precisely $(\sqrt{3})^2 = 3$
- To understand why this is true we must explore the world of finite-digit arithmetic

- Traditional mathematical world we permit numbers with an infinite number of digits.
- The arithmetic we use in this world *defines* √ 3 as that unique positive number that when multiplied by itself produces the integer 3.
- In the computational world, however, each representable number has only a fixed and finite number of digits.
- This means, for example, only most rational numbers can be represented exactly.
- √ 3 is not rational, approximate representation, will not be precisely 3, But sufficiently close to 3.
- The error that is produced when a calculator or computer is used to perform real number calculations is called **round-off error.**

Binary Machine Numbers

- A 64-bit (binary digit) representation is used for a real number. The first bit is a sign indicator, denoted *s. This is followed by an 11-bit exponent, c, called the* ***characteristic,*** and a 52-bit binary fraction, *f , called the* ***mantissa. The base for the exponent is 2.***
- Since 52 binary digits correspond to between 16 and 17 decimal digits, we can assume that a number represented in this system has at least 16 decimal digits of precision.
- The exponent of 11 binary digits gives a range of 0 to $2^{11}-1$ = 2047.
- So for positive and negative numbers become −1023 to 1024.

Binary Machine Numbers

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

Consider the machine number

$$0\ 10000000011\ 1011100100010000000000000000000000000000000000000000.$$

The leftmost bit is $s = 0$, which indicates that the number is positive. The next 11 bits, 10000000011, give the characteristic and are equivalent to the decimal number

$$c = 1 \cdot 2^{10} + 0 \cdot 2^9 + \cdots + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 = 1024 + 2 + 1 = 1027.$$

The exponential part of the number is, therefore, $2^{1027-1023} = 2^4$. The final 52 bits specify that the mantissa is

$$f = 1 \cdot \left(\frac{1}{2}\right)^1 + 1 \cdot \left(\frac{1}{2}\right)^3 + 1 \cdot \left(\frac{1}{2}\right)^4 + 1 \cdot \left(\frac{1}{2}\right)^5 + 1 \cdot \left(\frac{1}{2}\right)^8 + 1 \cdot \left(\frac{1}{2}\right)^{12}.$$

As a consequence, this machine number precisely represents the decimal number

$$(-1)^s 2^{c-1023}(1+f) = (-1)^0 \cdot 2^{1027-1023}\left(1+\left(\frac{1}{2}+\frac{1}{8}+\frac{1}{16}+\frac{1}{32}+\frac{1}{256}+\frac{1}{4096}\right)\right)$$

However, the next smallest machine number is

0 10000000011 1011100100001111111111111111111111111111111111111111,

and the next largest machine number is

0 10000000011 1011100100010000000000000000000000000000000000000001.

[27.5664062499999982236431605997495353221893310546875,

27.5664062500000017763568394002504646778106689453125).

The smallest normalized positive number that can be represented has $s = 0$, $c = 1$, and $f = 0$ and is equivalent to

$$2^{-1022} \cdot (1 + 0) \approx 0.22251 \times 10^{-307},$$

and the largest has $s = 0$, $c = 2046$, and $f = 1 - 2^{-52}$ and is equivalent to

$$2^{1023} \cdot (2 - 2^{-52}) \approx 0.17977 \times 10^{309}.$$

Numbers occurring in calculations that have a magnitude less than

$$2^{-1022} \cdot (1 + 0)$$

result in **underflow** and are generally set to zero. Numbers greater than

$$2^{1023} \cdot (2 - 2^{-52})$$

result in **overflow** and typically cause the computations to stop (unless the program has been designed to detect this occurrence). Note that there are two representations for the number zero; a positive 0 when $s = 0$, $c = 0$ and $f = 0$, and a negative 0 when $s = 1$, $c = 0$ and $f = 0$.

Decimal Machine Numbers

$$\pm 0.d_1 d_2 \ldots d_k \times 10^n, \quad 1 \le d_1 \le 9, \quad \text{and} \quad 0 \le d_i \le 9,$$

for each $i = 2, \ldots, k$. Numbers of this form are called k-digit *decimal machine numbers*.

Any positive real number within the numerical range of the machine can be normalized to the form

$$y = 0.d_1 d_2 \ldots d_k d_{k+1} d_{k+2} \ldots \times 10^n.$$

- The floating-point form of *y, denoted f l(y), is obtained by terminating the mantissa of y at k decimal digits.*

Decimal Machine Numbers

- The floating-point form of *y, denoted f l(y), is obtained by terminating the mantissa of y at k decimal digits.*
- *There are two common ways of performing this termination.*

1- **chopping, is to simply chop off the digits *dk+1dk+2 . . . . This produces the*** floating-point form

$$f\,l(y) = 0.d_1 d_2 \ldots d_k \times 10^n.$$

- 2- Rounding: adds $5 \times 10^{n-(k+1)}$ *to y and then chops the result to* obtain a number of the form

$$f\,l(y) = 0.\delta_1 \delta_2 \ldots \delta_k \times 10^n.$$

For rounding, when $d_{k+1} \geq 5$, we add 1 to d_k to obtain $f\,l(y)$; that is, we *round up*. When $d_{k+1} < 5$, we simply chop off all but the first k digits; so we *round down*. If we round down, then $\delta_i = d_i$, for each $i = 1, 2, \ldots, k$. However, if we round up, the digits (and even the exponent) might change.

Example 1 Determine the five-digit (a) chopping and (b) rounding values of the irrational number ***π***.

- *$\pi = 3.14159265...$*
- Written in normalized decimal form, we have
- *$\pi = 0.314159265... \times 10^1$.*
- **(a) The floating-point form of *π using five-digit chopping is***
- *$fl(\pi) = 0.31415 \times 10^1 = 3.1415$.*
- **(b) The sixth digit of the decimal expansion of *π is a 9, so the floating-point form of***
- *π using five-digit rounding is*
- *$fl(\pi) = (0.31415 + 0.00001) \times 10^1 = 3.1416$.*

Measuring approximation errors.

Suppose that p^* is an approximation to p. The **absolute error** is $|p - p^*|$, and the **relative error** is $\dfrac{|p - p^*|}{|p|}$, provided that $p \neq 0$. ■

Example 2 Determine the absolute and relative errors when approximating p by p^* when

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

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

(c) $p = 0.3000 \times 10^4$ and $p^* = 0.3100 \times 10^4$.

(a) For $p = 0.3000 \times 10^1$ and $p^* = 0.3100 \times 10^1$ the absolute error is 0.1, and the relative error is $0.333\overline{3} \times 10^{-1}$.

(b) For $p = 0.3000 \times 10^{-3}$ and $p^* = 0.3100 \times 10^{-3}$ the absolute error is 0.1×10^{-4}, and the relative error is $0.333\overline{3} \times 10^{-1}$.

(c) For $p = 0.3000 \times 10^4$ and $p^* = 0.3100 \times 10^4$, the absolute error is 0.1×10^3, and the relative error is again $0.333\overline{3} \times 10^{-1}$.

This example shows that the same relative error, $0.333\overline{3} \times 10^{-1}$, occurs for widely varying absolute errors. As a measure of accuracy, the absolute error can be misleading and the relative error more meaningful, because the relative error takes into consideration the size of the value. ■