

Ch 01-2: Errors

Dr. Feras Fraige

Round-off Errors and Computer Arithmetic

- The arithmetic performed by a calculator or computer is different from the arithmetic in algebra and calculus courses.
- $2+2 = 4$, $4 \cdot 8 = 32$, and $(\sqrt{3})^2 = 3$.
However, with computer arithmetic we expect exact results for $2+2 = 4$ and $4 \cdot 8 = 32$, 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* $\sqrt{3}$ 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.
- $\sqrt{3}$ 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 .

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

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

for each $i = 2, \dots, k$. 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

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

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

Decimal Machine Numbers

- The floating-point form of y , denoted $fl(y)$, 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 $d_{k+1}d_{k+2} \dots$. This produces the floating-point form

$$fl(y) = 0.d_1d_2 \dots d_k \times 10^n.$$

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

$$fl(y) = 0.\delta_1\delta_2 \dots \delta_k \times 10^n.$$

For rounding, when $d_{k+1} \geq 5$, we add 1 to d_k to obtain $fl(y)$; 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 $i = 1, 2, \dots, k$. 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 .

- $\pi = 3.14159265\dots$
- Written in normalized decimal form, we have
- $\pi = 0.314159265\dots \times 10^1$.
- **(a) The floating-point form of π using five-digit chopping is**
- $f l(\pi) = 0.31415 \times 10^1 = 3.1415$.
- **(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
- $f l(\pi) = (0.31415 + 0.00001) \times 10^1 = 3.1416$.

Measuring approximation errors.

Suppose that p^* is an approximation to p . The **absolute error** is $|p - p^*|$, and the **relative error** is $\frac{|p - p^*|}{|p|}$, 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 $0.333\bar{3} \times 10^{-1}$.
- (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 $0.333\bar{3} \times 10^{-1}$.
- (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 $0.333\bar{3} \times 10^{-1}$.

This example shows that the same relative error, $0.333\bar{3} \times 10^{-1}$, 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. ■