CSC349A Numerical Analysis

Lecture 4

Rich Little

University of Victoria

2023

Table of Contents I

Round-off errors

Round-off errors originate from two factors:

- finite representations of possibly infinitely long numbers
- finite range of values from a possibly infinite range

All dependent on the *word size* - maximum size of the string of bits used.

Number systems

- **Decimal:**
  - base-10
  - digits: 0,1,2,3,4,5,6,7,8,9
  - powers of 10 positional system
  - Ex: 86409

Number systems

- **Binary:**
  - base-2
  - digits: 0,1
  - powers of 2 positional system
  - Ex: 101011

Dec-Bin Conversion

University of Victoria Computer Science

- Convert 185_{10} to binary

Computer representation

- Positive integers
  - What we can represent depends on the word size
  - Ex: 8-bits, then $43_{10} = 00101011_2$
  - What is the range? $2^8 = 256$ values from 0 to $2^8 - 1$
  - Ex: 16-bits, then $43_{10} = 0000000000101011_2$
  - What is the range? $2^{16} = 65,536$ values from 0 to $2^{16} - 1$

Computer representation

- Negative integers
  - Signed magnitude method - use leftmost bit for the sign
  - usually 0 for '+' and 1 for '-'
  - Ex: 8-bits, then $-43_{10} = 10101011_2$
  - What is the range? only 7 bits so 128 values with a lead 0 and 128 with a lead 1
  - But, two of them - 10000000 and 00000000 - represent the same number, 0
  - We usually let 10000000 = -128 giving the range -128 to 127

Real numbers - Decimal

- How do we interpet real numbers in decimal?
- Ex: Consider 82.3801

Real numbers - Binary

- What about in binary?
- Ex: Consider 101.1101

Real numbers - Conversion

- Going from decimal to binary with real numbers?

- Now, how do we represent them in a computer?

Table of Contents I

Floating-point number system

- A floating-point number system is a finite approximation to the (infinite) real/complex number system, of the form:

$$\pm m \times b^e \tag{1}$$

- The m is called the **mantissa**, b is the **base** and e is the **exponent**.

Floating-point representation

- How do we store a floating-point number in a word?
- We take the component parts of the floating-point number, (sign, mantissa, exponent) and assign them to different sections of the word
- For example, take $\pm m \times b^e$ and put them into the word as follows:

s_m	s_e	exponent	mantissa

Normalized floating-point number system

- In normalized floating-point number systems real numbers are represented in the form:

$$\pm 0.d_1 d_2 d_3 \ldots d_k \times b^e \tag{2}$$

- Where the first digit of the *mantissa* should be non-zero, i.e. $1 \leq d_1 \leq b-1$.
- The remaining digits can be zero and are also constrained by the base, i.e. $0 \leq d_i \leq b-1$.
- Because $d_1 \neq 0$, k is the number of significant digits in the mantissa, and is called the *precision* of the floating point system.

Example - IEEE 754

- binary32 is the single-precision floating point representation
- it uses a 32-bit word with 1 bit for the sign of the mantissa, s_m, 8 bits for the signed (biased by 127) exponent, and the remaining 23 bits for the mantissa

s_m	exponent	mantissa
$1bit$	$8bits$	$23bits$

- this sytem has a precision of $k = 24$, don't need to store the lead 1

Errors in floating-point representation

There are a number of inherent errors in this system, some more obvious than others.

- Large negative and positive numbers fall outside the finite range of the system (overflow).
- Because of normalization, very small (close to 0) negative and positive numbers fall outside the range (underflow).
- Only a finite number of values can be represented in the range (round-off error)
- The distance between two consecutive floating-point numbers increases as the numbers get larger

Ex: Hypothetical Floating-Point Computer

Suppose your computer uses a 7-bit word to represent normalized floating-point numbers as follows:

s_m	s_e	e_1	e_2	b_1	b_2	b_3

where s_m is the sign of the mantissa and s_e is the sign of the exponent (0 for positive, 1 for negative), b_1, b_2, and b_3 are the bits of the mantissa, and e_1, e_2 are the bits of the exponent.

Example continued

(a) What is the smallest positive non-zero number that can be represented in this system? What is it's value in decimal?

(b) What is the next smallest positive non-zero number that can be represented in this system?

(c) What is the distance between the two values above? What does this value represent?

(d) Use (c) to predict the next value in decimal. Convert it to floating-point to test your answer.