These are the lecture notes for CSC349A Numerical Analysis taught by Rich Little. They roughly correspond to the material covered in each lecture in the classroom but the actual classroom presentation might deviate significantly from them depending on the flow of the course delivery. They are provide as a reference to the instructor as well as supporting material for students who miss the lectures. They are simply notes to support the lecture so the text is not detailed and they are not thoroughly checked. Use at your own risk.

1 Overview

The material covered in this lecture is partially based on handout 3. This material consists of the floating point number representation and round off errors.

2 Number systems

2.1 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.

2.2 Number systems

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

$$\begin{aligned} 86,409 &= (9 \times 10^0) + (0 \times 10^1) + (4 \times 10^2) + (6 \times 10^3) + (8 \times 10^4) \\ &= (9 \times 1) + (0 \times 10) + (4 \times 100) + (6 \times 1000) + (8 \times 10000) \\ &= 9 + 0 + 400 + 6000 + 80000 \\ &= 86409 \end{aligned}$$

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

$$\begin{aligned} 101011_2 &= (1 \times 2^0) + (1 \times 2^1) + (0 \times 2^2) + (1 \times 2^3) + (0 \times 2^4) + (1 \times 2^5) \\ &= (1 \times 1) + (1 \times 2) + (0 \times 4) + (1 \times 8) + (0 \times 16) + (1 \times 32) \\ &= 1 + 2 + 0 + 8 + 0 + 32 \\ &= 43_{10} \end{aligned}$$

2.3 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 $2^{16} - 1$
- 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

2.4 Real numbers

Back to decimal, how do we interpret real numbers? For example, what does 82.3801 represent?

$$\begin{aligned}82.3801 &= (8 \times 10^1) + (2 \times 10^0) + (3 \times 10^{-1}) + (8 \times 10^{-2}) + (0 \times 10^{-3}) + (1 \times 10^{-4}) \\ &= (8 \times 10) + (2 \times 1) + (3 \times 1/10) + (8 \times 1/100) + (0 \times 1/1000) + (1 \times 1/10000) \\ &= 80 + 1 + 0.3 + 0.08 + 0 + 0.0001 \\ &= 82.3801\end{aligned}$$

What about in binary? Consider 101.1101_2.

$$\begin{aligned}101.1101_2 &= (1 \times 2^2) + (1 \times 2^0) + (1 \times 2^{-1}) + (1 \times 2^{-2}) + (1 \times 2^{-4}) \\ &= 4 + 1 + 1/2 + 1/4 + 1/16 \\ &= 4 + 1 + 0.5 + 0.25 + 0.0625 \\ &= 5.8125_{10}\end{aligned}$$

Going from decimal to binary with real numbers? Now, how do we represent them in a computer?

3 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**. 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

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

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

Examples $+0.1234 \times 10^2$, -0.1111×2^4, three fingered alien floating-point number $+0.2222 \times 3^3$, not normalized 0.01111×10^2 should be instead 0.1111×10^1.

The first part is called the *mantissa*, b is the base and e is the *exponent*. The normalization refers to the property that 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. It is called the *precision* of the the floating point system. For example in a floating-point binary system each digit will be either 0 or 1 and in a deciaml floating-point system each digit will be between 0 and $9 = 10-1$. Common computer examples $b = 2, k = 24$ single precision, $b = 2, k = 53$ double precision.

3.1 Example 1 - IEEE 754

The IEEE 754 standard binary32 is a single-precision floating point representation for real numbers. 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 $k = 24$, because it doesn't need to store the lead 1.

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. Let's look at a hypothetical 7-bit computer to explore some of these ideas.

3.2 Example 2 - Hypothetical 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, s_e is the sign of the exponent, e_i are the bits of the exponent (an integer), and b_j are the bits of the mantissa (normalized), in the form

$$\pm 0.m_1m_2m_3 \times 2^{\pm e_1e_2}$$

Questions (a) What is the smallest positive, non-zero, value in this system?

$$+0.100_2 \times 2^{-11_2}$$

(b) What is it as a word?

0	1	1	1	1	0	0

Or, more simply

$$0111100_2$$

(c) What is it as a decimal value?

$$+0.100_2 \times 2^{-11_2} = 0.100_2 \times 2^{-3_{10}} = 1 \times 2^{-1} \times 2^{-3} = 1 \times 1/2 \times 1/8 = 1/16 = 0.0625_{10}$$

(d) What is the next smallest number?

$$\begin{aligned} 0111101_2 &= 0.101 \times 2^{-3} \\ &= (1/2 + 1/8) \times 1/8 \\ &= (0.5 + 0.125) \times 0.125 \\ &= 0.078125_{10} \end{aligned}$$

(e) What is the distance between these two values?

$$0.078125_{10} - 0.0625_{10} = 0.015625_{10}$$

This value is a measure of the error. This value is the same for every number in this system where the exponent is -3. However, it changes when the exponent changes.