Computational Physics

Numerical Accuracy

01/29/2009

Outline

Homework Assignment

1. Read Chapter 6
   - *“An Introduction to object-oriented analysis”*
2. Assignments (6) and (7) of Section 5.15
   - See handout!
   - For assignment 7, do not use the DISLIN package, instead save data for x, y, and the field to a file and plot in 3D using Gnuplot “splot”!
   - Due next Tuesday, February 3

Outline

Computing Numerics

- Representing Numbers
  - Bits and Bytes
  - Fixed Points (int or long)
  - Floating Points (float or double)
- Floating Point Arithmetic
- Computational Errors
  - Range Errors
  - Round-Off Errors

Outline

Representing Numbers

- Binary Bits
  - Units of Memory
  - All numbers eventually are represented in Binary Form
    ➜ Finite precision: *Limits & Approximation*
- Word Length
  - Number of bits to store a number (often given in Bytes)
    - 1 byte = 1 B = 8 bits = 8 b
    - 1 kB = 2^{10} bytes = 1024 bytes
    - 1 MB = 1024 $*$ 1024 bytes

Fixed Points (int or long)

- Exact Representation
- $2^N - 1$ integers represented by N bits
  - 1st bit gives the sign
  - Remaining $N - 1$ bits give the value
  - 32-bit integers:
    $-2\,147\,483\,648 \le \mathrm{int}_{32} \le 2\,147\,483\,648$
- *Integer Arithmetic* is exact!
  - Except for overflows and underflows

Bit Manipulation

```
# include <iostream.h>

main() {
    int thirdBit = (1 << 2);
    int i = 8;

    if ( (i & thirdBit) != 0 ) {
        cerr << endl << "Third bit true!";
    }
    else if ( (i & thirdBit) == 0 ) {
        cerr << endl << "Third Bit false!";
    }
}
```

1. Shift Operator <<
2. Bitwise Logical AND &

Floating Points (float or double)

- Scientific work mainly uses floating-point numbers
- Floating-Point Notation

	sign	exp + bias	mantissa
float_{32}	0	01111111	10000000000000000000000

$$x = (-1)^{\text{sign}} \cdot \text{mantissa} \cdot 2^{\text{exponent}}$$

1. mantissa: $1.m_1\, 2^{-1} + m_2\, 2^{-2} + \ldots$
2. bias = $0111\;1111_2 = 127_{10}$

Example: true exponent = $127 - 127 = 0$
➜ $(-1)^0 \cdot 2^0 \cdot 1.5 = 1.5$

Floating Points (float or double)

- Scientific work mainly uses floating-point numbers
- Floating-Point Notation

	sign	exp + bias	mantissa
float_{32}	0	10000000	00000000000000000000000

$$x = (-1)^{\mathrm{sign}} \cdot \mathrm{mantissa} \cdot 2^{\mathrm{exponent}}$$

1. mantissa: $1.m_1 2^{-1} + m_2 2^{-2} + \ldots$
2. bias = $0111\,1111_2 = 127_{10}$

Example: true exponent = 128 − 127 = 1
➜ $(-1)^0 \cdot 2^1 \cdot 1.0 = 2.0$

Floating Points (float or double)

- Scientific work mainly uses floating-point numbers
- Floating-Point Notation

	sign	exp + bias	mantissa
float_{32}	1	10000000	00000000000000000000000

$$x = (-1)^{\text{sign}} \cdot \text{mantissa} \cdot 2^{\text{exponent}}$$

1. mantissa: $1.m_1 2^{-1} + m_2 2^{-2} + \ldots$
2. bias = $0111\,1111_2 = 127_{10}$

Example: true exponent = 128 − 127 = 1
➜ $(-1)^1 \cdot 2^1 \cdot 1.0 = -2.0$

Floating Points (float or double)

- Scientific work mainly uses floating-point numbers
- Floating-Point Notation

	sign	exp + bias	mantissa
float_{32}	0	10000001	10000000000000000000000

$$x = (-1)^{\mathrm{sign}} \cdot \mathrm{mantissa} \cdot 2^{\mathrm{exponent}}$$

1. mantissa: $1.m_1 2^{-1} + m_2 2^{-2} + \ldots$
2. bias = $0111\,1111_2 = 127_{10}$

Example: true exponent = $129 - 127 = 2$

$\rightarrow (-1)^0 \cdot 2^2 \cdot 1.5 = 6.0$

Floating Points (float or double)

- Scientific work mainly uses floating-point numbers
- Floating-Point Notation

	sign	exp + bias	mantissa
float_{32}	0	01111110	10000000000000000000000

$$x = (-1)^{\text{sign}} \cdot \text{mantissa} \cdot 2^{\text{exponent}}$$

1. mantissa: $1.m_1 2^{-1} + m_2 2^{-2} + \ldots$
2. bias = $0111\,1111_2 = 127_{10}$

Example: true exponent = $126 - 127 = -1$
➜ $(-1)^0 \cdot 2^{-1} \cdot 1.5 = 0.75$

Floating Points (float or double)

- Scientific work mainly uses floating-point numbers
- Floating-Point Notation

	sign	exp + bias	mantissa
float_{32}	0	01111011	10011001100110011001101

$$x = (-1)^{\mathrm{sign}} \cdot \mathrm{mantissa} \cdot 2^{\mathrm{exponent}}$$

1. mantissa: $1.m_1 2^{-1} + m_2 2^{-2} + \ldots$
2. bias = $0111\,1111_2 = 127_{10}$

Example: true exponent = 123 − 127 = −4
➜ $(-1)^0 \cdot 2^{-4} \cdot 1.6 = 0.1$

Floating Points (float or double)

- Scientific work mainly uses floating-point numbers
- Floating-Point Notation

	sign	exp + bias	mantissa
float_{32}	0	01111111	10000000000000000000000

$$x = (-1)^{\text{sign}} \cdot \text{mantissa} \cdot 2^{\text{exponent}}$$

- *Floating-Point Arithmetic* is not exact!
  - Range

    $2.9 \cdot 10^{-39} \leq \text{float}_{32} \leq 3.4 \cdot 10^{38}$

    $10^{-322} \leq \text{double}_{64} \leq 10^{308}$

Outline

Computational Errors

- Human Errors
  - Blunders
- Random Errors
  - Acts of Nature
- Approximation Errors

$$e^x \approx \sum_{n}^{N} (-x)^n / n!$$

- Range Errors
- Round-Off Errors

Computational Errors

- Human Errors
  - Blunders
- Random Errors
  - Acts of Nature
- Approximation Errors

$$e^x \approx \sum_n^N (-x)^n / n!$$

- Range Errors
- Round-Off Errors

Range Errors

32 bit words

$$-2\,147\,483\,648 \leq \mathrm{int}_{32} \leq 2\,147\,483\,648$$

$$2.9 \cdot 10^{-39} \leq \mathrm{float}_{32} \leq 3.4 \cdot 10^{38}$$

If a number x is larger than the MAXVAL, an overflow occurs.

If a number x is smaller than the MINVAL, an underflow occurs.

The resulting value may be a NaN (not a number), a machine dependent value, or unpredictable.

Round-Off Errors

Machine Accuracy ϵ

The largest number such that

$$1.0 + \epsilon = 1.0$$

Computer Representation x_c:

$$x_c = x\,(1 + \epsilon_x) \quad |\epsilon_x| \leq \epsilon$$