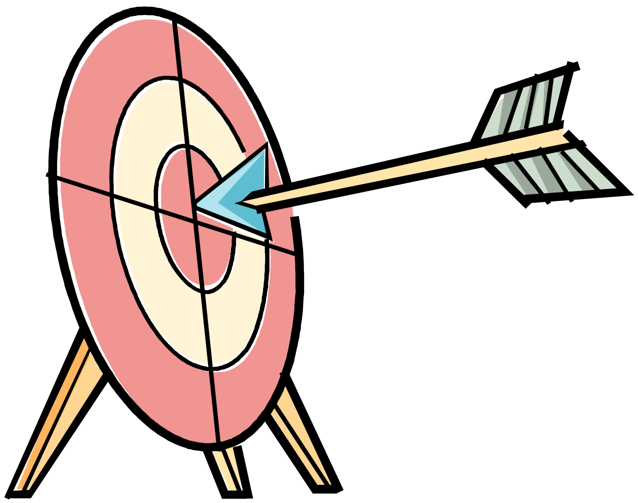
Machine



precision

and errors

Machine representation and precision

Every computer has a limit how small or large a number can be

A computer represent numbers in the binary form.

Word length: number of bytes used to store a number

Most common architecture:

Word length = 4 bytes = 32 bites

Word length = 8 bytes = 64 bites

(1 byte = 1 B = 8 bits: 00000000)

Integer numbers

For a 8 bit computer

1	1	1	1	1	1	1	1
2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

The highest number then: $2^8 - 1$ (–1 because the first is “0”)

Since we need 1 bit for +/-

Then the highest number is $2^7 - 1 = 127$

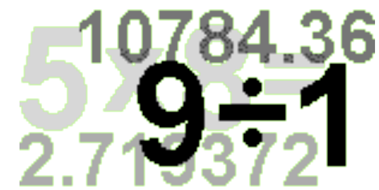
For 32-bit computers all integer numbers are in the range

$2^{31} - 1 = 2,147,483,647$

For N-bit computers the range is $[0,2^{N-1}]$

note: 1K = 1kB = 2^{10} bytes = 1024 bytes.

Floating point numbers



------------------------ Three blocks ----------------------------

0	1000 0000	1000000 0000 0000 0000 0000
signbit	8-bit exponent	23-bit mantissa

$$\boxed{x_{float} = (-1)^S \cdot mantissa \cdot 2^{\exp}}$$

range of exponent [-127,128] $(2^{128} \sim 10^{+38})$

Single precision : 6-7 decimal places $1/2^{23} \sim 1.2*10^{-7}$

range: max – about $\pm 3.402923 \times 10^{+38}$

range: min – about $\pm 1.401298 \times 10^{-45}$

machine precision ε: $1.0 + \varepsilon = 1.0$

Example

Getting a problem with the single precision is quite easy:

example – Bohr's radius

$$a_0 = \frac{4\pi\varepsilon_0\hbar^2}{m_e e^2} \approx 5.3 \cdot 10^{-11} m$$

where

$$\varepsilon_0 = 8.85 \cdot 10^{-12} C^2 / N \cdot m^2$$

$$\hbar = 6.63 \cdot 10^{-34} J \cdot s / 2\pi$$

$$m_e = 9.11 \cdot 10^{-31} kg$$

$$e = 1.60 \cdot 10^{-19} C$$

$1J = 1kg \cdot m^2 / s^2$ and $1N = 1kg \cdot m / s^2$

the numerator $1.24 \cdot 10^{-78}$

the denominator $2.33 \cdot 10^{-68}$

the single precision 10^{-38}

What to do?

- restructure the equation
- change units (scales)
- increase precision

Floating point: double precision (64 bits)

------------------------ Three blocks -----------------------------

signbit   11-bit exponent   52-bit mantissa

$$\boxed{x_{float} = (-1)^S \cdot mantissa \cdot 2^{\exp}}$$

range of exponent [-1023,1024]   ($2^{1024} \sim 10^{+308}$)

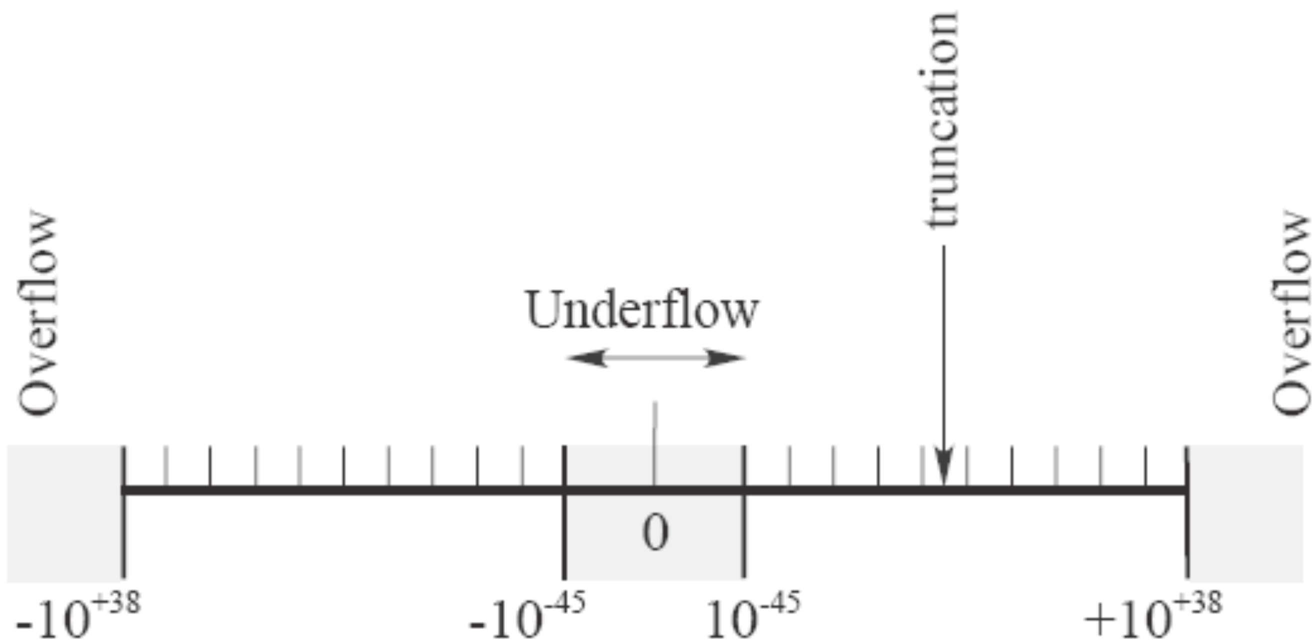
Double precision : 15-16 decimal places $1/2^{52} \sim 1.2*10^{-15}$

range: max – about $\pm 1.7976931348623157 \times 10^{+308}$

range: min – about $\pm 4.9406564584124654 \times 10^{-324}$

machine precision ε: $1.0 + \varepsilon = 1.0$

from “A Survey of Computational Physics. Introductory Computational Science” by R.Landau et al (2008)



```
// test on the machine precision
#include <iostream>
#include <iomanip>
#include <cmath>
using namespace std;
int main()
{
     float one, eps;
     eps = 1.0;
     for ( int j=1; j <=100; j++)
     {
         eps = eps/2.0;
         one = 1.0 + eps;
         cout << setw(5)<<j<< setiosflags(ios::scientific)
              << setw(12) << setprecision(6)  << eps
              << setiosflags(ios::fixed | ios::showpoint)
              << setw(15)<<setprecision(10)<< one << endl;
     } return 0;
}
```

```
21 4.76837e-07    1.000000477
22 2.38419e-07    1.000000238
23 1.19209e-07    1.000000119
24 5.96046e-08    1.000000000
25 2.98023e-08    1.000000000
```

```
// test on the machine precision
#include <iostream>
#include <iomanip>
#include <cmath>
using namespace std;
int main()
{
    double one, eps;
    eps = 1.0;
    for ( int j=1; j <=100; j++)
    {
          eps = eps/2.0;
        one = 1.0 + eps;
        cout << setw(5)<<j<< setiosflags(ios::scientific)
             << setw(12) << setprecision(6)  << eps
             << setiosflags(ios::fixed | ios::showpoint)
             << setw(23)<<setprecision(18)<< one << endl;
    }return 0;
}
```

```
   50 8.88178e-16    1.000000000000000089
   51 4.44089e-16    1.000000000000000044
   52 2.22045e-16    1.000000000000000022
   53 1.11022e-16    1.000000000000000000
   54 5.55112e-17    1.000000000000000000
```

Three Types of Errors*

1. Grammatical

   Using what is NOT in the programming language.

   The compiler finds them.

2. Errors in programming the algorithm

   Examples: (n-1) errors, inversion of logical tests, …

   We have to find them.

3. Mirabile visu (strange to behold) :

   They show up only for some input parameters (see Murphy's laws). Reasons: loss of significant digits (round off errors), iterative instabilities, … + many more. Developing good habits in programming helps to prevent these errors.

*Classification from F.S. Acton "Real Computing made real".

Type 3 - Typical errors

- Round off errors: any number is represented by a finite number of bits
- Approximation errors: from using approximations, like replacing

$\int_0^{\infty} f(x)dx$ on $\int_0^{a} f(x)dx$ with finite *a*

Round off Errors

Loss of significant digits

Example: 3.1425926 – 3.1425811 = 0.0000115

we have lost five significant digits in a single subtraction!

(Multiplications and divisions with real numbers do not lose significant digits.)

Loss of precision

Erosion by repeated rounding errors (the least significant digits being eroded away first.)

The average accumulated multiplication error after N steps is about

$$\varepsilon_N \approx \sqrt{N}\varepsilon_0$$

more examples:

example 1

$$f(x) = \frac{(1-x^2)}{(1-x)}$$ for $x \to 1$ would produce a “noisy” result

but after reducing the expression $f(x) = (1+x)$

example 2

$(a^2 - b^2) = (a-b)*(a+b)$ for $a \to b$

Loss of significant digits

Lost of significant digits “occur in so many ways that they defy useful classification and lack systematic cures.”

Always use double precision for calculations in physics research

Approximation errors

Dealing with infinity:

Solutions 1: transform variables.

Example for integration:

$$\int_a^{\infty} f(x)dx \qquad \int_{-\infty}^{\infty} f(x)dx$$

Transform variable of integration so that new interval is finite: y=1/(x+1) (but: not to introduce singularities)

Approximation errors

Dealing with infinity:

Solution 2: work with finite numbers, but evaluate "tails".

Example: use the asymptotic behavior

$$\int_0^\infty \frac{\sqrt{x}}{x^2+1}dx = \int_0^a \frac{\sqrt{x}}{x^2+1}dx + \int_a^\infty \frac{\sqrt{x}}{x^2+1}dx$$

for a >> 1

$$\int_a^\infty \frac{\sqrt{x}}{x^2+1}dx = \int_a^\infty \frac{1}{x^{3/2}}dx = \frac{2}{\sqrt{a}}$$

Random errors

ECC memory

Error-Correcting Code memory, a type of memory that includes special circuitry for detecting and correcting system memory errors by adding additional bits and using special codes.

In the ECC code each data signal corresponds to specific rules. Departures from these rules in the receiver can be automatically detected and corrected.

Blunders

Only two things are infinite, the universe and human stupidity, and I'm not sure about the former.
Albert Einstein