生態模擬： 以 C 語言為例

Class 11

2018/06/07

How to solve algebraic equation and how to reduce numerical errors

Takeshi Miki

三木　健（海洋研究所）

11.1 Numerical errors (0): How to use complex number

You can use complex number including the library <complex.h>

Just try to compile and execute ‘complex_test0.c’

```
#include <stdio.h>
#include <math.h>
#include <stdlib.h>
#include <time.h>
#include <complex.h>

int main(void){

    double complex x;

    x = 3.0 + I*4.0;

     printf("real=%lf\timg=%lf\tabsolute value = %lf\n", creal(x), cimag(x), cabs(x));

     return 0;
}
```

Real part c

Imaginary

length of

Functions in real **sqrt(x), exp(x), fabs(x)…** ←→ **csqrt(x), cexp(x), cabs(x)…**

11.2 Numerical errors (0): IEEE 754

IEEE 754: Standard for Binary Floating-Point Arithmetic, which is organized the **I**nstitute of **E**lectrical and **E**lectronics **E**ngineers, Inc.) (I triple E).

We need many roles for arithmetic calculations in computer for keeping the quality of calculation with controlling errors.
IEEE 754 is adopted as default specification in some language (e.g. C#, Java) but not in C language. However, it is recommended to follow IEEE754 even in C language.

For example, IEEE 754 determines the specification of **float** and **double**.
→**Topic 1**

Also, it determines the roles of **rounding algorithms** (c.f. 四舍五入)
→**Topic 2**

11.2 Numerical errors (0): IEEE 754, floating point number

What is floating point number?

In the decimal system (十進位)

-10.34 $= - 1.\boxed{034 \times 10}^{1}$

fraction

exponent

sign

base

*significant digit = 4

In the binary system (二進位)

$-2.5 = -(1\times2^{1} + 0\times2^{0} + 1\times2^{-1}) = -10.1$

-10.1 $= - 1.01 \times 2^{1}$

11.2 Numerical errors (0): IEEE 754, floating point number

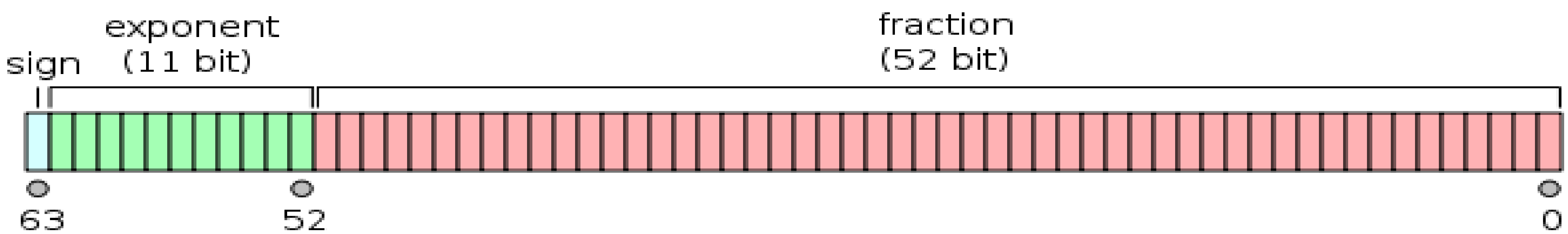
"double" is defined in IEEE 754

In the binary system (二進位)

$-2.5 = -(1\text{x}2^{1} + 0\text{x}2^{0} + 1\text{x}2^{-1}) = -10.1$

$$-10.1 = \boldsymbol{-\ 1.01 \times 2^{1}}$$

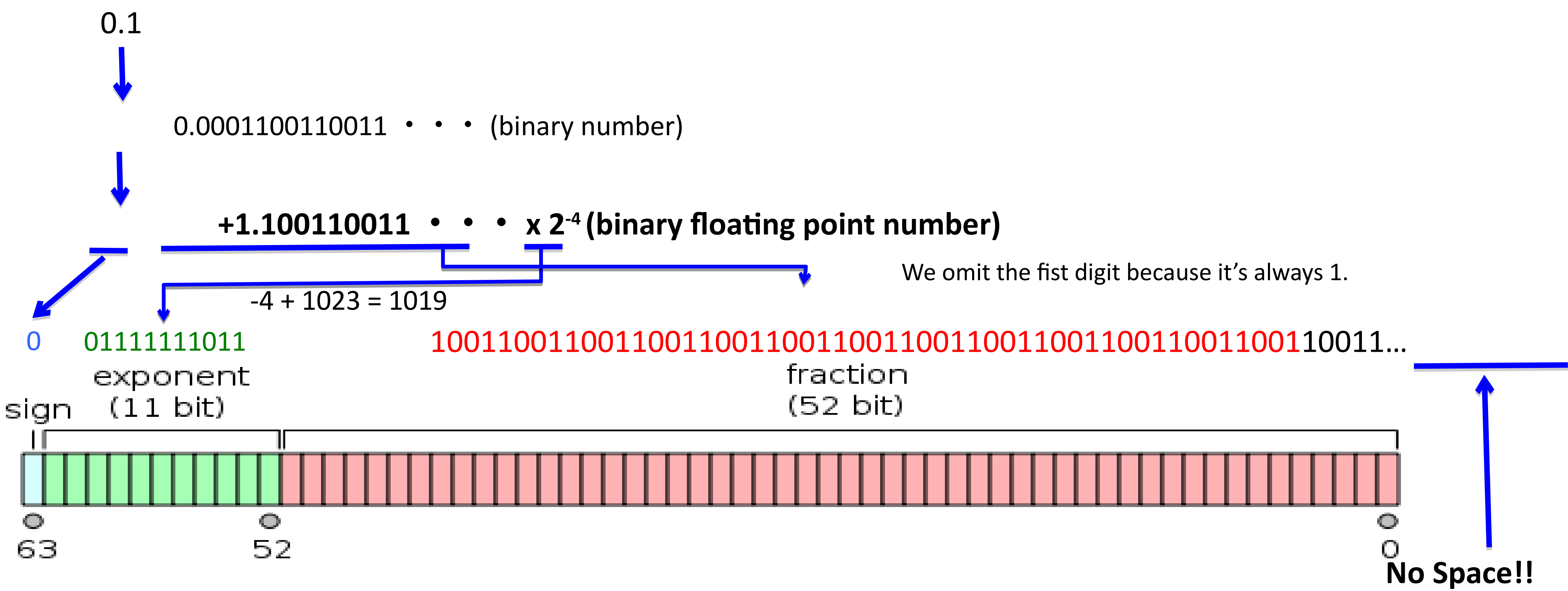
Type Double (64bit)



http://ja.wikipedia.org/wiki/

What is the source of rounding 'error'??

Consider converting 0.1 (in decimal system) to a binary number and to floating point number. We need one digit for 0.1 in decimal system, but in the binary system we need infinite number of digits: 0.1 = 0.0001100110011 • • • .
However, we can use only 52 bit of memory to stock the fraction part of this number.



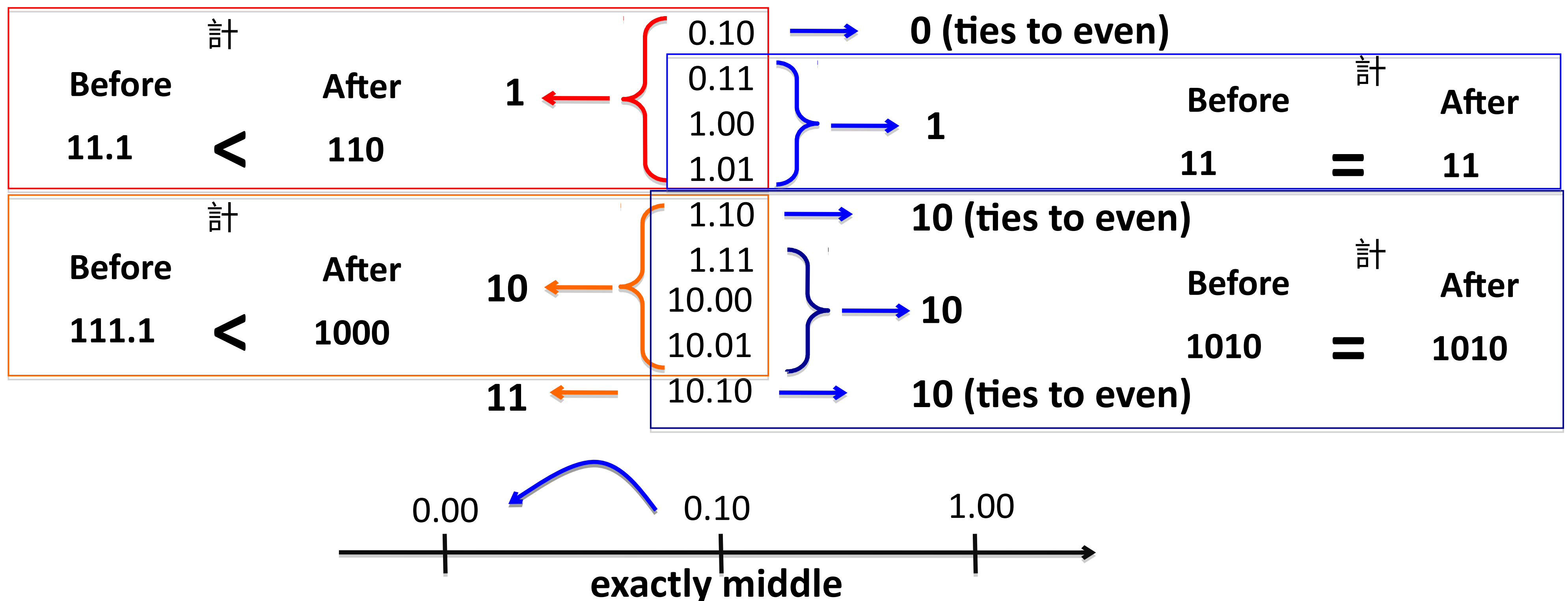
We need common rules on how to round numbers.

One of common rules in IEEE 754 is ‘**Roundings to nearest, ties to even**’

Consider a simple hypothetical trial to round two digits of binary for understanding the difference bet. the **roundings to nearest** and simple **0 舍 1 入**

0 舍 1 入

‘Roundings to nearest, ties to even’



11.2 Numerical errors (0): IEEE 754, rounding algorithms

Common rules in IEEE 754 is not included in mathematical library <math.h>.

One of alternative libraries is:

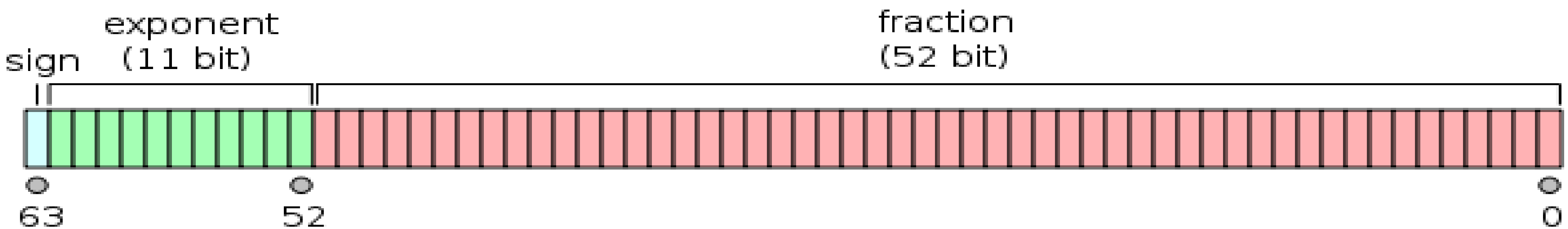
CRlibm Correctly Rounded mathematical library

http://lipforge.ens-lyon.fr/www/crlibm/

However, I would not say Crlibm is always better than libm…

cf) intel C compiler (icc) also includes specific math library

Do you think this type of error is critical?



11.3 Numerical errors (1): what is the loss of significant digits?

Let's consider a simple example, trying to calculate the following when the significant digit number = 8 (let's think only in the decimal system).

$$\sqrt{1001} - \sqrt{999}$$

Round-off errors occur in the final digit.

$$\sqrt{1001} = 31.6385840\cdots \approx 31.63858\underline{4} \qquad \text{(error: } 1.2 \times 10^{-7}\ \%\text{)}$$

$$\sqrt{999} = 31.6069612\cdots \approx 31.60696\underline{1} \qquad \text{(error: } 8.2 \times 10^{-7}\ \%\text{)}$$

$$\sqrt{1001} - \sqrt{999} \approx 31.638584 - 31.606961$$

$$= \underline{0.0}3162\underline{3} = 3.162\underline{3000}e-2$$

← should be 3.1622781e-2

error: 6.9 x 10^{-4} %!!

Round-off error in the final digit.

Round-off error jumps up to 5th digit!!

→ loss of significant digits!!

11.4 Numerical errors (2): what happens in logistic map?

Let's consider a simulation of logistic map, using x as float or y as double (see logistic_error.c)**.**

```
float x = 0.1;
double y = 0.1;
```

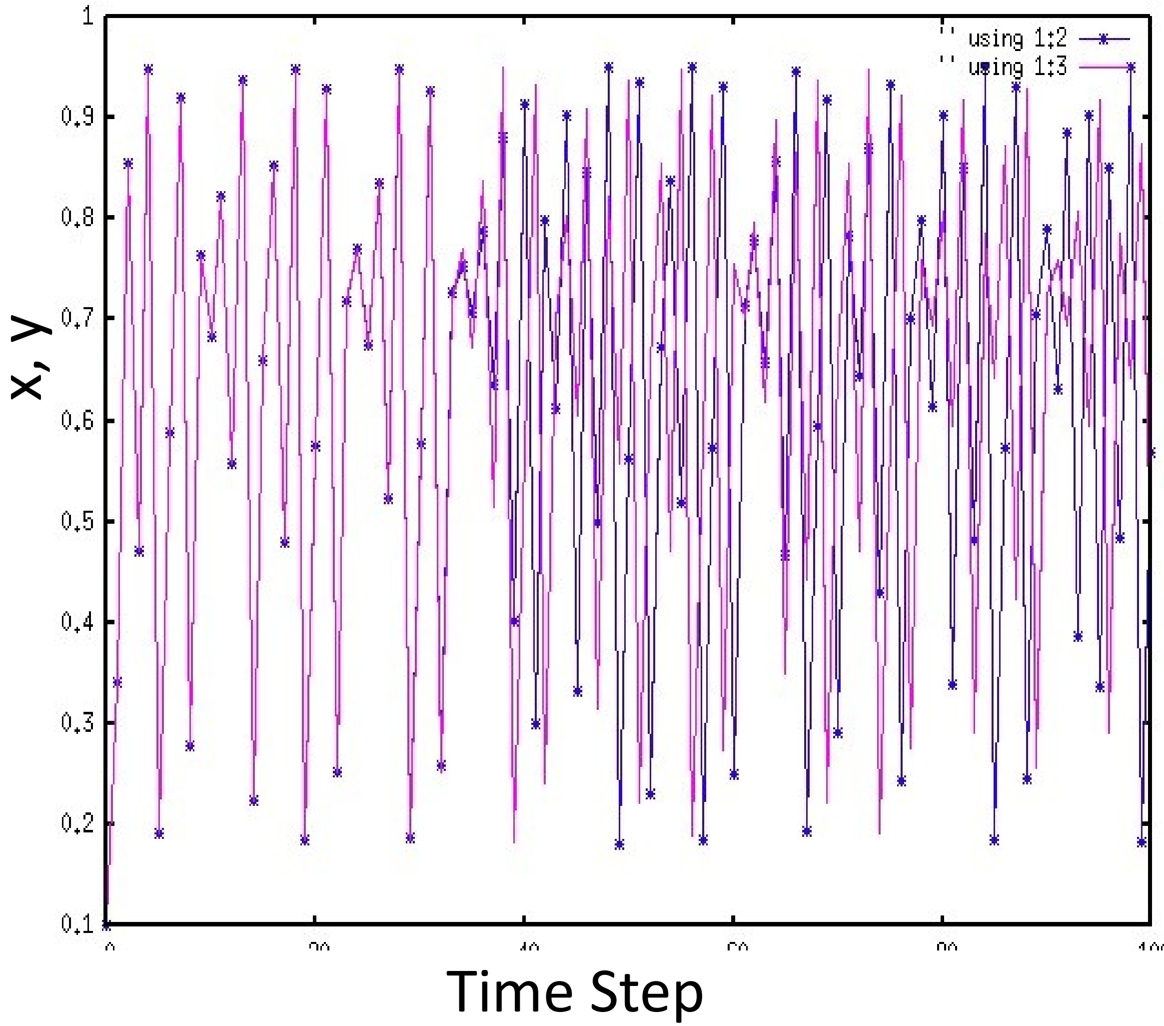
$$x_{t+1} = 3.8 \cdot x_t \cdot (\underline{1.0 - x_t})$$

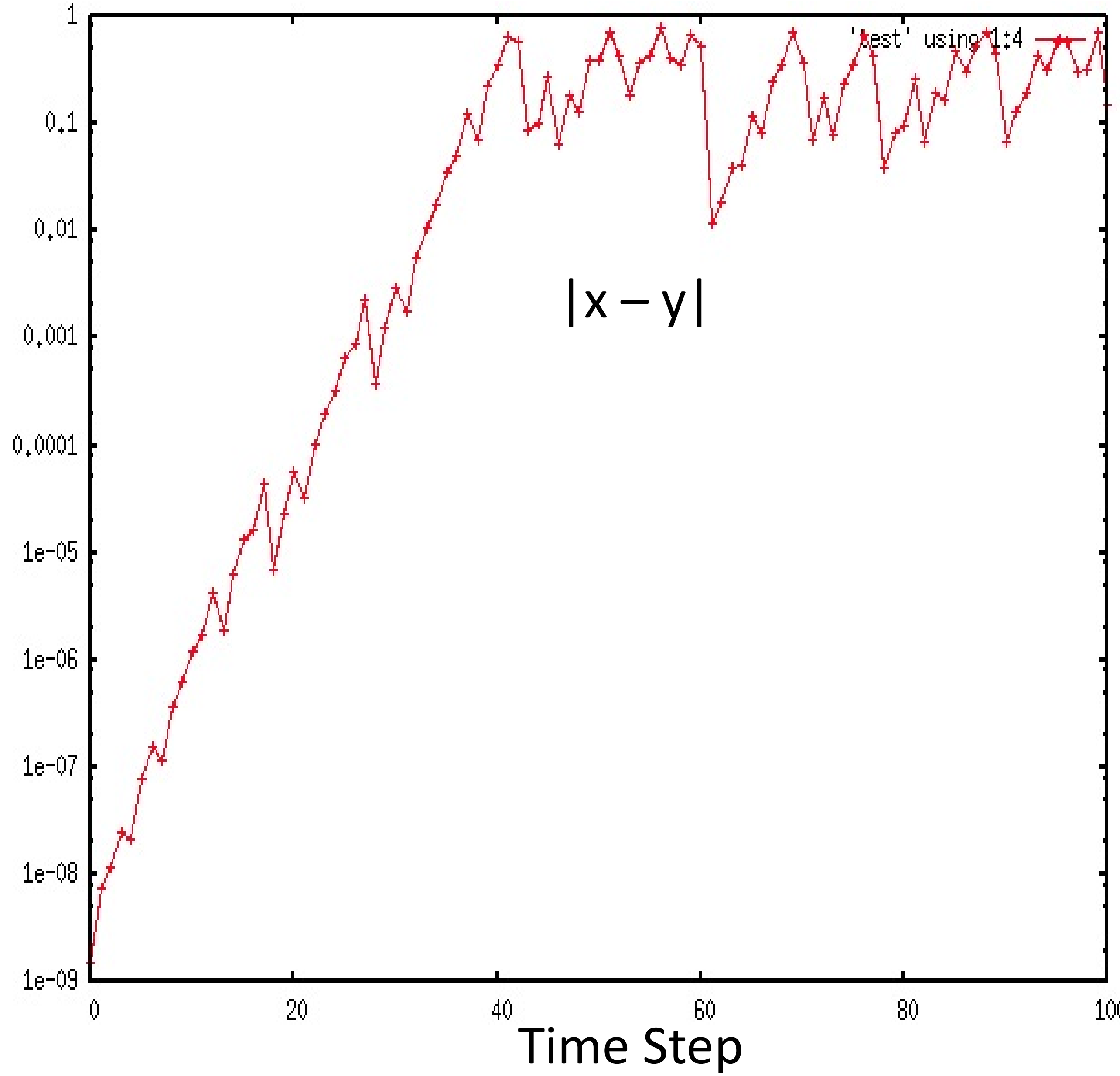
$$y_{t+1} = 3.8 \cdot y_t \cdot (\underline{1.0 - y_t})$$

Loss of significant digits occurs
In this subtraction.

11.4 Numerical errors (2): what happens in logistic map?

```
float x = 0.1;
double y = 0.1;
```





Suggestions from logistic map:

The source of the cancellation of significant digit is the rounding error.

Therefore, we can suppress this error by using *double* instead of *float*.

→ In some computer systems, you can further increase the memory size for floating point numbers.

Is this unique solution?

→NO!!

11.5 Numerical errors (3): how to solve quadratic equation

First , we need to go back to…

$$\sqrt{1001} - \sqrt{999} \approx 31.638584 - 31.606961$$

$$= 0.031623 = 3.1623000\mathrm{e}-2$$

← should be 3.16227805e-2

error: 6.9 x 10^{-4} %!!

We can reduce the error by avoiding the subtraction between similar size of number!!

$$\sqrt{1001} - \sqrt{999} = \frac{(\sqrt{1001} - \sqrt{999})(\sqrt{1001} + \sqrt{999})}{(\sqrt{1001} + \sqrt{999})}$$

$$= \frac{2}{(\sqrt{1001} + \sqrt{999})} \approx \frac{2}{31.638684 + 31.606961}$$

$$= \frac{2}{63.245545} \approx 0.031622781$$

error: 1.8 x 10^{-6} %!!

11.5 Numerical errors (3): how to solve quadratic equation

Then, try to think about the good algorithm for solving quadratic equation.

We have an formula for exact solution !!

$$ax^2 + bx + c = 0$$

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

But, when $b^2 >> 4ac$

$$-b \pm \sqrt{b^2 - 4ac} \approx -b \pm |b|$$

…leading to loss of significant digits

11.5 Numerical errors (3): how to solve quadratic equation

Let's assume $D = b^2 - 4ac > 0$

A better formula is:

$$\text{if} \quad (b > 0)$$

$$x_1 = \frac{-b - \sqrt{b^2 - 4ac}}{2a}$$

$$\text{else}$$

$$x_1 = \frac{-b + \sqrt{b^2 - 4ac}}{2a}$$

$$x_2 = \frac{c}{a x_1}$$

11.5 Numerical errors (3): how to solve quadratic equation

Compare the performance of *quadratic_solver* and *quadratic_bad_solver* in solver_test0.c for following equations.

$$(x-1)(x-5)=0$$

$$(x-1)(x-1.001)=0$$

$$(x-1.0\cdot 10^{6})(x-1.2345678912345\cdot 10^{-10})=0$$

$$x^2+x+1=0$$

11.6 Further information

Generally, there are gaps between analytical formula and numerical algorithm.

In calculations of floating point number, the following 'associative law' does NOT hold.

$$a + (b + c) = (a + b) + c$$

'Numerical Verification method' to control the worst error size and thus ensure the quality of simulations is a growing field in applied math. With this method, we are able to prove mathematical theorems using computer!! e.g. existence or uniqueness of solution or attractor in ODE/PDE

http://www.econ.hit-u.ac.jp/~finmodel/events/pdf/Kobayashi07-24-10.pdf