Chapter 1
Finite precision arithmetic

Floating point representation

Let us consider

$$2013.9 = 2 \cdot 10^3 + 0 \cdot 10^2 + 1 \cdot 10^1 + 3 \cdot 10^0 + 9 \cdot 10^{-1} = (2013.9)_{10}.$$

In general, a real number x is expressed as

$$\begin{aligned} x &= \pm(d_n d_{n-1} \cdots d_1 d_0 . d_{-1} d_{-2} \cdots)_\beta \\ &= \pm\left(d_n \beta^n + d_{n-1}\beta^{n-1} + \cdots + d_1\beta^1 + d_0\beta^0 + d_{-1}\beta^{-1} + d_{-2}\beta^{-2} + \cdots\right), \end{aligned}$$

where β is the base, d_i ($0 \le d_i \le \beta - 1$) are digits. The number system with $\beta = 10$ is called the decimal number system.

Example 1. The number system with $\beta = 2$ is called the binary number system.

$$(1010.1)_2 = 1 \cdot 2^3 + 1 \cdot 2^1 + 1 \cdot 2^{-1} = 8 + 2 + 0.5 = (10.5)_{10}.$$

Example 2.

$$(0.2)_{10} = (0.00110011\ldots)_2.$$

In a computer, the floating point representation is used. That is, a real number x with n significant digits is expressed as

$$x = \pm(0.d_1 d_2 \cdots d_n)_\beta \cdot \beta^e, \quad d_1 \neq 0, \quad -M \le e \le M.$$

Here, $(0.d_1 d_2 \cdots d_n)_\beta$ is the mantissa (or significand), and e is the exponent.

Example 3. For $x = 0.2$ with $n = 2$ and $\beta = 2$, we have

$$(0.11)_2 \cdot 2^{-2} = 0.1875.$$

Fall 2013 Math 471 Sec 2
Introduction to Numerical Methods

Manabu Machida (University of Michigan)

Example 4. Let us consider the largest and smallest positive real numbers $x_{\max}, x_{\min}$ that a computer with $\beta = 2$, $n = 4$, $M = 3$ can express.

$$x_{\max} = (0.1111)_2 \cdot 2^3 = (2^{-1} + 2^{-2} + 2^{-3} + 2^{-4}) \cdot 2^3 = 2^2 + 2 + 1 + 2^{-1} = 7.5,$$
$$x_{\min} = (0.1000)_2 \cdot 2^{-3} = 2^{-1} \cdot 2^{-3} = 2^{-4} = 0.0625.$$

In IEEE[1] double precision format, each number is stored as a string of 64 bits[2].

$\pm$	$\pm$	mantissa $= 52$ bits	exponent $= 10$ bits

The first two bits are for the signs of the mantissa and exponent. Hence we have $\beta = 2$, $n = 52$, and $M = (1111111111)_2 = 2^{10} - 1 = 1023$. Note that $2^{1023} \approx 10^{308}$.

Roundoff error

If x is a real number and $\mathrm{fl}(x)$ is its floating point representation, then $x - \mathrm{fl}(x)$ is the roundoff error.

Example 5. Let us consider π.

$$\begin{aligned}\pi &= 3.14159265358979\ldots \\ &= 2 + 1 + \frac{1}{8} + \frac{1}{64} + \frac{1}{4096} + \cdots = 2^1 + 2^0 + 2^{-3} + 2^{-6} + 2^{-12} + \cdots \\ &= (11.001001000001\ldots)_2 = (0.11001001000001\ldots)_2 \cdot 2^2\end{aligned}$$

If $n = 4$, then $\mathrm{fl}(\pi) = (0.1101)_2 \cdot 2^2 = 3.25$. This is the closest 4-bit floating point number to π. With $n = 52$, the roundoff error in $\mathrm{fl}(\pi)$ is approximately $2^{-52} \cdot 2^2 \approx 10^{-15}$.

Subtraction often causes loss of significance. That is, the result has fewer significant digits.

Example 6. Let us consider $\sqrt{0.01523} = 0.12340988\ldots$ and $\sqrt{0.01521} = 0.12332882\ldots$. We have

$$\sqrt{0.01523} - \sqrt{0.01521} = 0.000081057405\ldots.$$

With 4 significant digits, we obtain

$$\sqrt{0.01523} - \sqrt{0.01521} = 0.1234 - 0.1233 = 0.0001 = (0.1000)_{10} \cdot 10^{-3}.$$

Now the result has only 1 significant digit.

Example 7. Let us consider the quadratic formula for $ax^2 + bx + c = 0$:

[1] Institute of Electrical and Electronics Engineers

[2] A bit (**bi**nary digi**t**) is a digit in the base 2. A byte is 8 bits, and so 64 bits (b) are 8 bytes (B).

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

Using a computer with $n = 4$ (that is, each arithmetic step is rounded to 4 digits), suppose you write a code to solve quadratic equations by the quadratic formula. Let us find the solutions to $0.2x^2 - 47.91x + 6 = 0$. We have

$$\begin{aligned}
x &= \frac{47.91 \pm \sqrt{47.91^2 - 4(0.2)6}}{2(0.2)} = \frac{47.91 \pm \sqrt{2295 - 4.8}}{0.4} = \frac{47.91 \pm \sqrt{2290}}{0.4} \\
&= \frac{47.91 \pm 47.85}{0.4} = \begin{cases} \dfrac{47.91 + 47.85}{0.4} = \dfrac{95.76}{0.4} = 239.4, \\[2mm] \dfrac{47.91 - 47.85}{0.4} = \dfrac{0.06}{0.4} = 0.15. \end{cases}
\end{aligned}$$

All 4 digits of 239.4 are correct. However, only 1 digit is correct for 0.15. We can improve the algorithm as follows.

$$\begin{aligned}
x &= \frac{-b - \sqrt{b^2 - 4ac}}{2a} \cdot \frac{-b + \sqrt{b^2 - 4ac}}{-b + \sqrt{b^2 - 4ac}} = \frac{b^2 - (b^2 - 4ac)}{2a(-b + \sqrt{b^2 - 4ac})} = \frac{2c}{-b + \sqrt{b^2 - 4ac}} \\
&= \frac{2 \cdot 6}{47.91 + 47.85} = \frac{12}{95.76} = 0.1253.
\end{aligned}$$

Now all 4 digits are correct.

Finite-difference approximation

We consider finite-difference approximation of a derivative. By recalling the definition of the derivative of a function $f(x)$, we have

$$f'(x) \approx \frac{f(x+h) - f(x)}{h} = D_+ f(x),$$

where h is a small positive number. We call D_+ the forward finite-difference operator. To estimate the error, we consider the Taylor series:

$$f(x) = f(a) + f'(a)(x-a) + \frac{1}{2} f''(a)(x-a)^2 + \cdots.$$

By $x \to x + h$, $a \to x$, we obtain

$$f(x+h) = f(x) + f'(x)h + \frac{1}{2} f''(x)h^2 + \cdots.$$

Thus,

$$\frac{f(x+h)-f(x)}{h} = f'(x) + \underbrace{\frac{1}{2}f''(x)h + \cdots}_{\text{truncation error}} .$$

Hence the error is proportional to h. We write this as

$$D_+ f(x) = f'(x) + O(h),$$

where the symbol $O(h)$ means “order h”: $O(h) = ch + O(h^2)$.

Example 8. For $f(x) = \mathrm{e}^x$, we numerically compute $f'(1)$. The exact value is $f'(1) = \mathrm{e} = 2.71828\ldots$.

h	$D_+ f$	$\lvert f' - D_+ f\rvert$	$\lvert f' - D_+ f\rvert / h$
0.1	2.8588	0.1406	1.4056
0.05	2.7874	0.0691	1.3821
0.025	2.7525	0.0343	1.3705
0.0125	2.7353	0.0171	1.3648
$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$
0	e	0	$\frac{1}{2}f''(1) = \mathrm{e}/2$

Let us investigate the error by writing the following Matlab code. The result is shown in Fig. 1.1. If error $\approx ch^p$, then p is called the order of accuracy of the approximation. Since $\log(\text{error}) \approx \log(ch^p) = \log c + p \log h$, the slope of the data on the log-log plot is p. We see that $p = 1$ for not too small h.

```
exact_value=exp(1);
for j=1:65
  h(j)=1/2^(j-1);
  computed_value=(exp(1+h(j))-exp(1))/h(j);
  error(j)=abs(exact_value-computed_value);
end
% log-log plot
loglog(h,error,h,error,'o'); xlabel('h'); ylabel('error');
```

The computed value has two sources of error: *truncation error* is due to replacing the exact derivative $f'(x)$ by the finite-difference approximation $D_+ f(x)$, and *roundoff error* is due to using finite precision arithmetic.

The above example shows that the error is not necessarily small for very small h. The truncation error is $O(h)$ and the roundoff error is $O(\varepsilon/h)$, where $\varepsilon \approx 10^{-15}$ in Matlab. Thus the total error is $O(h) + O(\varepsilon/h)$. Hence, for large h the truncation error dominates the roundoff error, but for small h the roundoff error dominates the truncation error.

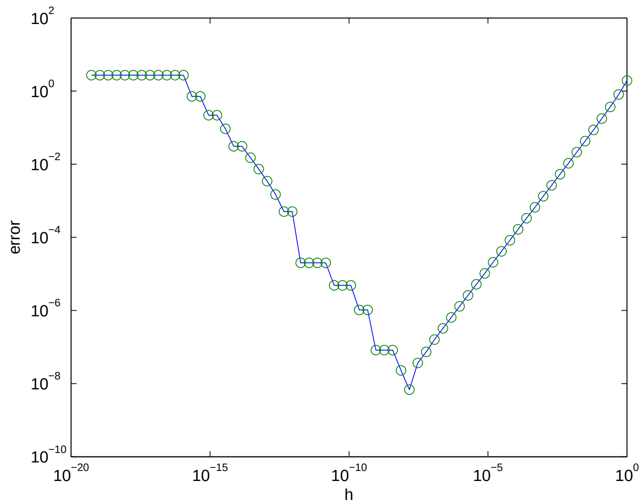

Fig. 1.1 Error of $D_{+}e^{x}$ at $x = 1$ as a function of h.