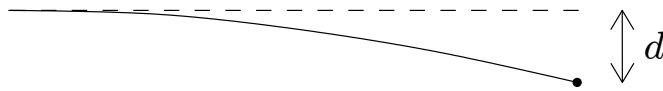


section 1.3, 1.4 : finite precision arithmetic

ex : page 31, “An Example to Set the Stage”

$E = \frac{4mgl^3}{dab^3}$  : modulus of elasticity of a metal beam (rectangular cross-section)



$m = 0.491 \pm 0.0005$  kg : mass of object suspended from tip of beam

$g = 9.81$  m/s<sup>2</sup> : acceleration due to gravity

$l = 0.451 \pm 0.0005$  m : length

$d = 0.142 \pm 0.0005$  m : deflection

$a = 0.021 \pm 0.0005$  m : width

$b = 0.003 \pm 0.0005$  m : thickness

max value of  $E = \frac{4(0.4915)(9.81)(0.4515)^3}{0.1415(0.0205)(0.0025)^3} \frac{\text{kg} \cdot \text{m}/\text{s}^2 \cdot \text{m}^3}{\text{m} \cdot \text{m} \cdot \text{m}^3} = 39.165 \times 10^9 \text{ N}/\text{m}^2$

min value of  $E = \frac{4(0.4905)(9.81)(0.4505)^3}{0.1425(0.0215)(0.0035)^3} \dots\dots\dots = 13.397 \times 10^9 \text{ N}/\text{m}^2$

This shows that small errors in  $m, l, d, a, b$  can lead to a large error in  $E$ .

number systems

$$x = \pm(d_n d_{n-1} \cdots d_1 d_0 . d_{-1} d_{-2} \cdots)_\beta$$

$$= \pm(d_n \beta^n + d_{n-1} \beta^{n-1} + \cdots d_1 \beta^1 + d_0 \beta^0 + d_{-1} \beta^{-1} + d_{-2} \beta^{-2} \cdots)$$

$\beta$  : base ,  $d_i$  : digits ,  $0 \leq d_i \leq \beta - 1$

ex

$\beta = 10$  : decimal

$$(2008)_{10} = 2 \cdot 10^3 + 0 \cdot 10^2 + 0 \cdot 10^1 + 8 \cdot 10^0$$

$\beta = 2$  : binary

$$(10101.01)_2 = 1 \cdot 2^4 + 0 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 + 0 \cdot 2^{-1} + 1 \cdot 2^{-2}$$

$$= 16 + 4 + 1 + 0.25 = 21.25 = 2 \cdot 10^1 + 1 \cdot 10^0 + 2 \cdot 10^{-1} + 5 \cdot 10^{-2} = (21.25)_{10}$$

Computers use the following floating point representation of a real number.

$x = \pm(0.d_1d_2 \cdots d_n)_\beta \cdot \beta^e$  ,  $d_1 \neq 0$  , we say there are  $n$  significant digits

$(0.d_1d_2 \cdots d_n)_\beta$  : mantissa ,  $e$  : exponent ,  $-M \leq e \leq M$

ex

consider a computer with  $\beta = 2$  ,  $n = 4$  ,  $M = 3$

$$x_{\min} = (0.1000)_2 \cdot 2^{-3} = (0.0001)_2 = 2^{-4} = (0.0625)_{10}$$

$$x_{\max} = (0.1111)_2 \cdot 2^3 = (111.1)_2 = 2^2 + 2^1 + 2^0 + 2^{-1} = (7.5)_{10}$$

note

1. In IEEE double precision format, each number is stored in memory as a string of 64 bits.

		<b>mantissa</b>	<b>exponent</b>
--	--	-----------------	-----------------

The first two bits are sign bits for the mantissa and exponent, the next 52 bits are for the mantissa, and the remaining 10 bits are for the exponent. Hence we have  $\beta = 2, n = 52, M = 2^{10} = 1024$ .

2. If  $x$  is a real number and  $\text{fl}(x)$  is its floating point representation, then  $x - \text{fl}(x)$  is the roundoff error. The IEEE standard gives rules for determining  $\text{fl}(x)$ .

ex

$$\pi = 3.14159265358979 \dots = 2 + 1 + \frac{1}{8} + \frac{1}{64} + \frac{1}{4096} + \dots = (11.00100100001 \dots)_2$$

$$n = 4 \Rightarrow \text{fl}(\pi) = (11.01)_2 = 3.25 : \text{closest floating point number to } \pi$$

In reality, with  $n = 52$ , the roundoff error in  $\text{fl}(\pi)$  is about  $2^{-52} \approx 10^{-15}$ .

note

If two floating point numbers with  $n$  significant digits are subtracted, the result may have fewer than  $n$  significant digits. This is called loss of significance due to cancellation.

ex

$$0.1234 - 0.1233 = 0.0001 : \text{the result has only 1 significant digit}$$

ex : page 45, “The Quadratic Formula”

$$ax^2 + bx + c = 0 \Rightarrow x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

$$0.2x^2 - 47.91x + 6 = 0 \Rightarrow x = 239.4247, 0.1253 : \text{Matlab}$$

Now suppose we use 4 decimal digit arithmetic.

$$\begin{aligned} x &= \frac{47.91 \pm \sqrt{47.91^2 - 4(0.2)6}}{2(0.2)} = \frac{47.91 \pm \sqrt{2295 - 4.8}}{0.4} = \frac{47.91 \pm \sqrt{2290}}{0.4} \\ &= \frac{47.91 \pm 47.85}{0.4} = \begin{cases} \frac{47.91 + 47.85}{0.4} = \frac{95.76}{0.4} = 239.4 & : \text{all 4 digits are correct} \\ \frac{47.91 - 47.85}{0.4} = \frac{0.06}{0.4} = 0.15 & : \text{only 1 digit is correct} \end{cases} \end{aligned}$$

The problem occurs because  $\sqrt{b^2 - 4ac} \approx b$  and this led to loss of significance in the subtraction  $47.91 - 47.85$ . One remedy is to use higher precision arithmetic (Matlab), but another remedy is to reformulate the arithmetic.

$$\begin{aligned} x &= \frac{-b - \sqrt{b^2 - 4ac}}{2a} \cdot \frac{-b + \sqrt{b^2 - 4ac}}{-b + \sqrt{b^2 - 4ac}} = \frac{b^2 - (b^2 - 4ac)}{2a(-b + \sqrt{b^2 - 4ac})} = \frac{2c}{-b + \sqrt{b^2 - 4ac}} \\ &= \frac{2 \cdot 6}{47.91 + 47.85} = \frac{12}{95.76} = 0.1253 : \text{now all 4 digits are correct} \end{aligned}$$

ex : finite-difference approximation of a derivative

$$f'(x) \approx \frac{f(x+h) - f(x)}{h} = D_+f(x) : \text{forward difference approximation}$$

question : how large is the error?

$$\text{Taylor series} : f(x) = f(a) + f'(a)(x-a) + \frac{1}{2}f''(a)(x-a)^2 \dots$$

$$\left. \begin{array}{l} x \rightarrow x+h \\ a \rightarrow x \end{array} \right\} \Rightarrow f(x+h) = f(x) + hf'(x) + \frac{h^2}{2}f''(x) + \dots$$

$$\Rightarrow f'(x) = \frac{f(x+h) - f(x)}{h} - \underbrace{\frac{h}{2}f''(x) + \dots}_{\text{error}}$$

$\uparrow$                        $\uparrow$                        $\uparrow$   
 exact                  approximation                  error  
 value

Hence the error is proportional to  $h$ ; we write this as  $f'(x) = D_+f(x) + O(h)$ , where the symbol  $O(h)$  means “of order  $h$ ”.

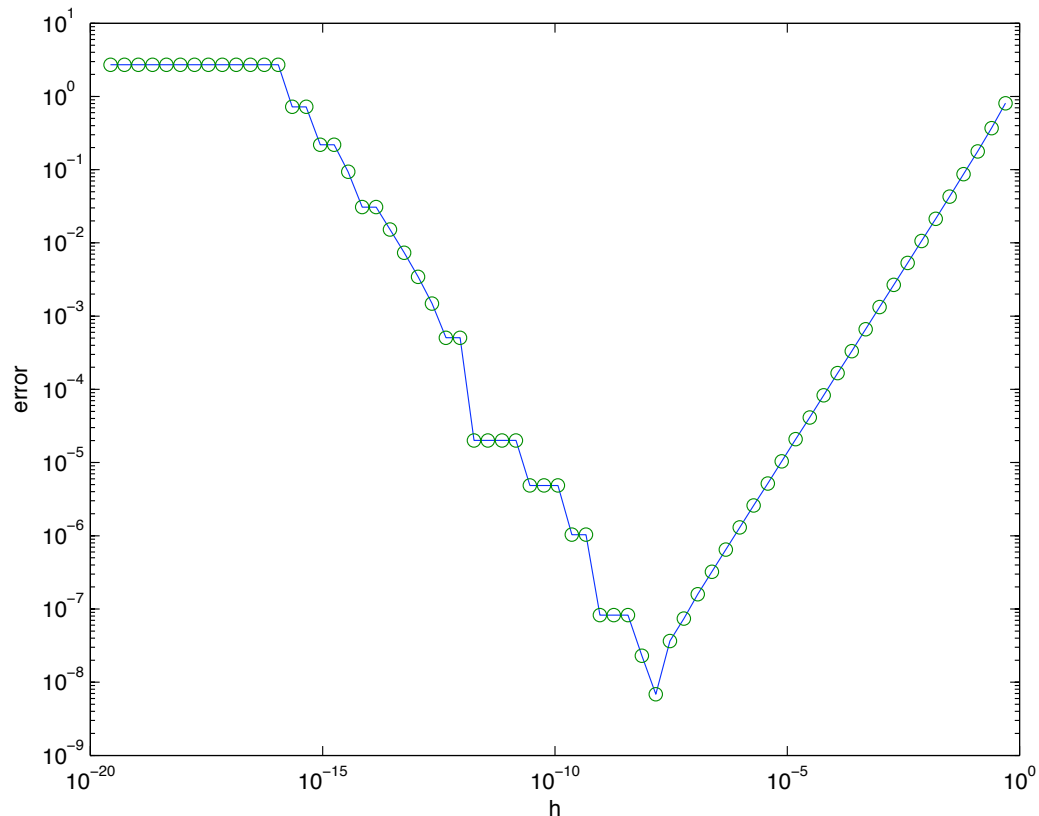
For example, if  $f(x) = e^x$ ,  $x = 1$ , then  $f'(1) = e = 2.71828\dots$  is the exact value.

$h$	$D_+f$	$f'(x) - D_+f$	$(f'(x) - D_+f)/h$
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
↓	↓	↓	↓
0	$e$	0	$-\frac{e}{2} = -\frac{1}{2}f''(1)$

note : In practice, something unexpected happens when  $h$  is small.

% finite-difference approximation of a derivative in Matlab

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



question : Why does the error increase for small  $h$ ?

1. 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 computer arithmetic.
2. The truncation error is  $O(h)$  and the roundoff error is  $O(\epsilon/h)$ , where  $\epsilon \approx 10^{-15}$  in Matlab.
3. For large  $h$ , the truncation error dominates the roundoff error, but for small  $h$ , the roundoff error dominates the truncation error.

note

$$D_-f(x) = \frac{f(x) - f(x-h)}{h} \quad : \quad \text{backward difference}$$

$$D_0f(x) = \frac{f(x+h) - f(x-h)}{2h} \quad : \quad \text{centered difference} \quad (\text{hw1})$$