section 1.3, 1.4 : finite precision arithmetic

number systems

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

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

β : base , d_i : digits , $0 \le d_i \le \beta - 1$

ex

$\beta = 10$: decimal

$(2009)_{10} = 2 \cdot 10^3 + 0 \cdot 10^2 + 0 \cdot 10^1 + 9 \cdot 10^0$

$\beta = 2$: binary

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

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

question: how are real numbers represented in a computer?

answer: floating point representation

$x = \pm(0.d_1 d_2 \cdots d_n)_\beta \cdot \beta^e$, $d_1 \neq 0$, we say there are n significant digits

$(0.d_1 d_2 \cdots d_n)_\beta$: mantissa , e : exponent , $-M \le e \le M$

ex

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

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

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

note

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

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 $\beta = 2, n = 52, M = (1111111111)_2 = 2^{10} - 1 = 1023$.

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

$$\pi = 3.14159265358979\ldots = 2 + 1 + \tfrac{1}{8} + \tfrac{1}{64} + \tfrac{1}{4096} + \cdots$$
$$= (11.001001000001\ldots)_2 = (0.11001001000001\ldots)_2 \cdot 2^2$$

$n = 4 \Rightarrow \text{fl}(\pi) = (0.1101)_2 \cdot 2^2 = 3.25$: closest 4-bit floating point number to π

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

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 of digits.

ex : $0.1234 - 0.1233 = 0.0001 = (0.1000)_{10} \cdot 10^{-3} \Rightarrow$ the result has only 1 significant digit

ex : page 45, quadratic formula

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

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

Now suppose we use 4 decimal digit arithmetic.

$$x = \frac{47.91 \pm \sqrt{47.91^2 - 4(0.2)6}}{2(0.2)} = \frac{47.91 \pm \sqrt{2295 - 4.8}}{0.4} = \frac{47.91 \pm \sqrt{2290}}{0.4}$$

$$= \frac{47.91 \pm 47.85}{0.4} = \begin{cases} \dfrac{47.91 + 47.85}{0.4} = \dfrac{95.76}{0.4} = 239.4 & : \text{ all 4 digits are correct} \\ \dfrac{47.91 - 47.85}{0.4} = \dfrac{0.06}{0.4} = 0.15 & : \text{ only 1 digit is correct} \end{cases}$$

The problem is due to loss of significance in the subtraction $47.91 - 47.85$. One remedy is to use higher precision arithmetic (Matlab), but another option is to reformulate the arithmetic.

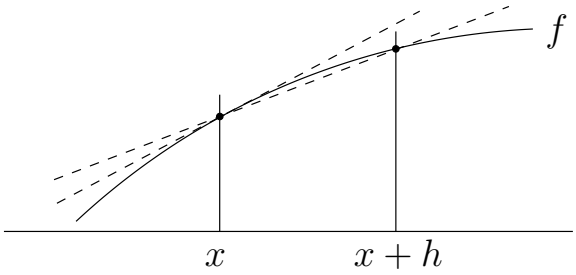
$$x = \frac{-b - \sqrt{b^2 - 4ac}}{2a} \cdot \frac{-b + \sqrt{b^2 - 4ac}}{-b + \sqrt{b^2 - 4ac}} = \frac{b^2 - (b^2 - 4ac)}{2a(-b + \sqrt{b^2 - 4ac})} = \frac{2c}{-b + \sqrt{b^2 - 4ac}}$$

$$= \frac{2 \cdot 6}{47.91 + 47.85} = \frac{12}{95.76} = 0.1253 \ : \text{ now all 4 digits are correct}$$

ex : finite-difference approximation of a derivative

forward difference

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



question : how large is the error?

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

equivalent form :

$$\left.\begin{array}{l} x \to x+h \\ a \to x \end{array}\right\} \Rightarrow \quad f(x+h) = f(x) + f'(x)h + \frac{1}{2}f''(x)h^2 + \cdots$$

$$\Rightarrow \quad f'(x) = \frac{f(x+h) - f(x)}{h} \underbrace{- \frac{h}{2}f''(x) + \cdots}$$

$$\begin{array}{ccc} \uparrow & \uparrow & \uparrow \\ \text{exact value} & \text{approximation} & \text{discretization error} \end{array}$$

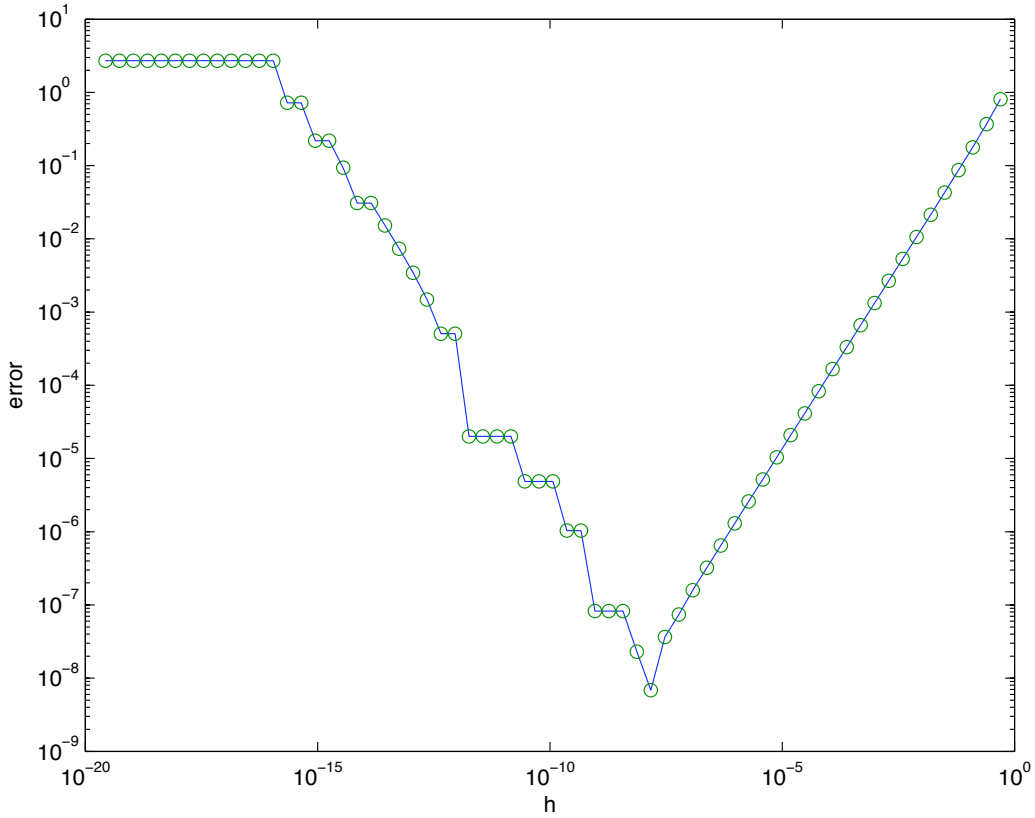
Hence the error is proportional to h; we write this as $f'(x) = D_+ f(x) + O(h)$, where the symbol $O(h)$ means "order h".

For example, if $f(x) = e^x, x = 1$, then $f'(1) = e = 2.71828\ldots$ 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
$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$
0	e	0	$-\frac{e}{2} = -\frac{1}{2}f''(1)$

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

```
% Matlab
  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(exact_value - computed_value);
  end
  loglog(h,error,h,error,'o');
  xlabel('h'); ylabel('error');
```



note

If error $\approx ch^p$, then $\log(\text{error}) \approx \log c + p \log h$, i.e. the slope of the data gives the order of convergence (this is why we use log-log to plot the error vs. h).

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. The total error is $O(h) + O(\epsilon/h)$. Hence, for large h, the truncation error dominates the roundoff error, but for small h, the roundoff error dominates the truncation error.

note

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

$$D_0 f(x) = \frac{f(x+h) - f(x-h)}{2h} \quad : \quad \text{centered difference} \quad \text{(hw)}$$