Chapter 1

Computer Arithmetic

1.1 Floating-point number systems

Definition 1.1. A *bit* is the basic unit of information in computing; it can have only one of two values 0 and 1.

Definition 1.2. A *byte* is a unit of information in computing that commonly consists of 8 bits; it is the smallest addressable unit of memory in many computers.

Definition 1.3. A *word* is a group of bits with fixed size that are handled as a unit by the instruction set architecture (ISA) and/or hardware of the processor. The *word size/width/length* is the number of bits in a word and is an important characteristic of processor or computer architecture.

Example 1.1. 32-bit and 64-bit computers are mostly common these days. A 32-bit register can store 2^{32} values, hence a processor with 32-bit memory address can directly access 4GB byte-addressable memory.

Definition 1.4 (Floating point numbers)**.** A *floating point number* (FPN) is a number of the form

$$x = \pm m \times \beta^e, \tag{1.1}$$

where $e \in [L, U]$ and the *significand* (or *mantissa*) m has the form

$$m = \left(d_0 + \frac{d_1}{\beta} + \cdots + \frac{d_{p-1}}{\beta^{p-1}} \right), \tag{1.2}$$

where the integer d_i satisfies $\forall i \in [0, p-1]$, $d_i \in [0, \beta-1]$. d_0 and d_{p-1} are called the *most significant digit* and the *least significant digit*, respectively. The portion $.d_1 d_2 \cdots d_{p-1}$ is called the *fraction*.

Algorithm 1.5. A decimal integer can be converted to a binary number via the following method:

- divide by 2 and record the remainder,
- repeat until you reach 0,
- concatenate the remainder backwards.

A decimal fraction can be converted to a binary number via the following method:

- multiply by 2 and check whether the integer part is greater than 1: if so record 1; otherwise record 0,
- repeat until you reach 0,
- concatenate the recorded bits forward.

Combine the above two methods and we can convert any decimal number to its binary counterpart.

Example 1.2. Convert 156 to binary number:

$$156 = (10011100)_2.$$

Example 1.3. What is the normalized binary form of $\frac{2}{3}$?

$$\begin{aligned}\frac{2}{3} &= (0.a_1 a_2 a_3 \cdots)_2 = (0.1010\cdots)_2 \\ &= (1.0101010\cdots)_2 \times 2^{-1}.\end{aligned}$$

Definition 1.6 (FPN systems)**.** A *floating point number system* $\mathcal{F}$ is a proper subset of the rational numbers $\mathbb{Q}$, and it is characterized by a 4-tuple (β, p, L, U) with

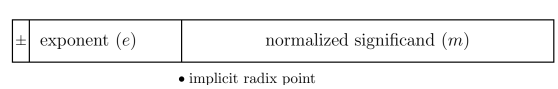
- the *base* (or radix) β;
- the *precision* (or significand digits) p;
- the *exponent range* $[L, U]$.

Definition 1.7. An FPN is *normalized* if its mantissa satisfies $1 \le m < \beta$.

Definition 1.8 (IEEE standard 754-2008)**.** The *single precision and double precision* FPNs of current IEEE (Institute of Electrical and Electronics Engineers) standard 754 are normalized FPN systems with the respective characterizations,

$$\beta = 2,\ p = 23 + 1,\ e \in [-126, 127], \tag{1.3a}$$
$$\beta = 2,\ p = 52 + 1,\ e \in [-1022, 1023]. \tag{1.3b}$$

Example 1.4. IEEE 754 has some further details.



(a) Out of the 32 bits, 1 is reserved for the sign, 8 for the exponents, 23 for the significand (see the plot above for the locations and the implicit radix point).

(b) The precision is 24 because we can choose $d_0 = 1$ for normalized binary floating point numbers and get away with never storing d_0.

(c) The exponent has $2^8 = 256$ possibilities. If we assign $1, 2, \ldots, 256$ to these possibilities, it would not be possible to represent numbers whose magnitudes are smaller than one. Hence we subtract $1, 2, \ldots, 256$ by 128 to shift the exponents to $-127, -126, \ldots, 0, \ldots, 127, 128$. Out of these numbers in the 2008 standard, $\pm m \times \beta^{-127}$ is reserved for ± 0 and $\pm m \times \beta^{128}$ is reserved for any number with a magnitude too large to be representable by the FPN system.

Definition 1.9. The *machine precision* of a normalized FPN system $\mathcal{F}$ is the distance between 1.0 and the next larger FPN in $\mathcal{F}$,

$$\epsilon_M := \beta^{1-p}. \tag{1.4}$$

Definition 1.10. The underflow limit (UFL) and the overflow limit (OFL) of a normalized FPN system $\mathcal{F}$ are respectively

$$\mathrm{UFL}(\mathcal{F}) := \min|\mathcal{F} \setminus \{0\}| = \beta^L, \tag{1.5}$$

$$\mathrm{OFL}(\mathcal{F}) := \max|\mathcal{F}| = \beta^U(\beta - \beta^{1-p}). \tag{1.6}$$

Example 1.5. By default matlab adopts IEEE 754 double precision arithmetic. Three characterizing constants are

- `eps` is the machine precision

$$\epsilon_M = \beta^{1-p} = 2^{1-(52+1)} = 2^{-52} \approx 2.22 \times 10^{-16},$$

- `realmin` is UFL($\mathcal{F}$)

$$\min|\mathcal{F} \setminus \{0\}| = \beta^L = 2^{-1022} \approx 2.22 \times 10^{-308},$$

- `realmax` is OFL($\mathcal{F}$)

$$\max|\mathcal{F}| = \beta^U(\beta - \beta^{1-p}) \approx 1.80 \times 10^{308}.$$

Corollary 1.11 (Cardinality of $\mathcal{F}$)**.** For a normalized binary FPN system $\mathcal{F}$,

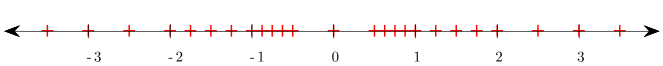
$$\#\mathcal{F} = 2^p(U - L + 1) + 1. \tag{1.7}$$

Proof. The cardinality can be proved by Axiom 0.11. The factor 2^p comes from the sign bit and the mantissa. By Example 1.4, $U-L+1$ is the number of exponents represented in $\mathcal{F}$. The trailing "+1" in (1.7) accounts for the number 0. □

Definition 1.12. The *range* of a normalized FPN system is a subset of $\mathbb{R}$,

$$\mathcal{R}(\mathcal{F}) := \{x : x \in \mathbb{R}, \mathrm{UFL}(\mathcal{F}) \le |x| \le \mathrm{OFL}(\mathcal{F})\}. \tag{1.8}$$

Example 1.6. Consider a normalized FPN system with the characterization $\beta = 2, p = 3, L = -1, U = +1$.



The four FPNs

$$1.00 \times 2^0,\ 1.01 \times 2^0,\ 1.10 \times 2^0,\ 1.11 \times 2^0$$

correspond to the four ticks in the plot starting at 1 while

$$1.00 \times 2^1,\ 1.01 \times 2^1,\ 1.10 \times 2^1,\ 1.11 \times 2^1$$

correspond to the four ticks starting at 2.

Definition 1.13. Two normalized FPNs a, b are *adjacent* to each other in $\mathcal{F}$ iff

$$\forall c \in \mathcal{F} \setminus \{a, b\}, \quad |a - b| < |a - c| + |c - b|. \tag{1.9}$$

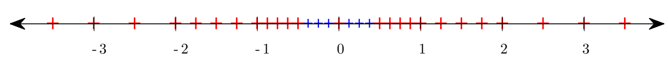
Lemma 1.14. Let a, b be two adjacent normalized FPNs satisfying $|a| < |b|$ and $ab > 0$. Then

$$\beta^{-1}\epsilon_M|a| < |a - b| \le \epsilon_M|a|. \tag{1.10}$$

Proof. Consider $a > 0$, then $\Delta a := b - a > 0$. By Definitions 1.4 and 1.7, $a = m \times \beta^e$ with $1.0 \le m < \beta$. a and b only differ from each other at the least significant digit, hence $\Delta a = \epsilon_M \beta^e$. Since $\frac{\epsilon_M}{\beta} < \frac{\epsilon_M}{m} \le \epsilon_M$ and thus $\frac{\Delta a}{a} \in (\beta^{-1}\epsilon_M, \epsilon_M]$. The other case is similar. □

Definition 1.15. The *subnormal* or *denormalized* numbers are FPNs of the form (1.1) with $e = L$ and $m \in (0, 1)$. A normalized FPN system can be *extended* by including the subnormal numbers.

Example 1.7. Add subnormal FPNs to the FPN system in Example 1.6 and we have the following plot.



1.2 Rounding error analysis

1.2.1 Rounding a single number

Definition 1.16 (Rounding)**.** *Rounding* is a map $\mathrm{fl} : \mathbb{R} \to \mathcal{F} \cup \{\mathrm{NaN}\}$. The default rounding mode is *round to nearest*, i.e. $\mathrm{fl}(x)$ is chosen to minimize $|\mathrm{fl}(x) - x|$ for $x \in \mathcal{R}(\mathcal{F})$. In the case of a tie, $\mathrm{fl}(x)$ is chosen by *round to even*, i.e. $\mathrm{fl}(x)$ is the one with an even last digit d_{p-1}.

Definition 1.17. A rounded number $\mathrm{fl}(x)$ *overflows* if $|x| > \mathrm{OFL}(\mathcal{F})$, in which case $\mathrm{fl}(x)$=NaN, or *underflows* if $0 < |x| < \mathrm{UFL}(\mathcal{F})$, in which case $\mathrm{fl}(x)$=0. An underflow of an extended FPN system is called a *gradual underflow.*

Definition 1.18. The *unit roundoff* of $\mathcal{F}$ is the number

$$\epsilon_u := \frac{1}{2}\epsilon_M = \frac{1}{2}\beta^{1-p}. \tag{1.11}$$

Lemma 1.19. For rounding to nearest, the unit roundoff for an FPN system with precision $p + k$ is $\beta^{p-1-k}\epsilon_u\epsilon_M$.

Proof. According to Definitions 1.16 and 1.18, the unit roundoff for an FPN system with precision $p + k$ is

$$\frac{1}{2}\beta^{1-p-k} = \frac{1}{2}\beta^{1-p}\beta^{1-p}\beta^{p-1-k} = \beta^{p-1-k}\epsilon_u\epsilon_M,$$

where the last step follows from Definitions 1.9 and 1.18. □

Theorem 1.20. For $x \in \mathcal{R}(\mathcal{F})$ as in (1.8), we have

$$\mathrm{fl}(x) = x(1+\delta), \qquad |\delta| < \epsilon_u. \tag{1.12}$$

Proof. By Definition 0.18, $\mathcal{R}(\mathcal{F})$ is a subset of $\mathbb{R}$ and is thus a chain. Therefore $\forall x \in \mathcal{R}(\mathcal{F})$, $\exists x_L, x_R \in \mathcal{F}$ s.t.

- x_L and x_R are adjacent,
- $x_L \le x \le x_R$.

If $x = x_L$ or x_R, then $\mathrm{fl}(x) - x = 0$ and (1.12) clearly holds. Otherwise $x_L < x < x_R$. Then Lemma 1.14 and Definitions 1.13 and 1.16 yield

$$|\mathrm{fl}(x)-x| \le \frac{1}{2}|x_R - x_L| \le \epsilon_u \min(|x_L|, |x_R|) < \epsilon_u |x|. \tag{1.13}$$

Hence $-\epsilon_u|x| < \mathrm{fl}(x) - x < \epsilon_u|x|$, which yields (1.12). □

Theorem 1.21. For $x \in \mathcal{R}(\mathcal{F})$, we have

$$\mathrm{fl}(x) = \frac{x}{1+\delta}, \qquad |\delta| \le \epsilon_u. \tag{1.14}$$

Proof. The proof is the same as that of Theorem 1.20, except that we replace the last inequality "$< \epsilon_u|x|$" in (1.13) by "$\le \epsilon_u|\mathrm{fl}(x)|$." Consequently, the equality in (1.14) holds when $x = \frac{1}{2}(x_L + x_R)$ and $\mathrm{fl}(x) = x_L$ has $m = 1.0$. □

Exercise 1.8. Find x_L, x_R of $x = \frac{2}{3}$ in normalized single-precision IEEE 754 standard, which of them is $\mathrm{fl}(x)$?

1.2.2 Binary floating-point operations

Definition 1.22 (Addition/subtraction of two FPNs)**.** Express $a, b \in \mathcal{F}$ as $a = M_a \times \beta^{e_a}$ and $b = M_b \times \beta^{e_b}$ where $M_a = \pm m_a$ and $M_b = \pm m_b$. With the assumption $|a| \ge |b|$, the sum $c := \mathrm{fl}(a+b) \in \mathcal{F}$ is calculated in a register of precision at least $2p$ as follows.

(i) Exponent comparison:
- If $e_a - e_b > p + 1$, set $c = a$ and return c;
- otherwise set $e_c \leftarrow e_a$ and $M_b \leftarrow M_b/\beta^{e_a - e_b}$.

(ii) Perform the addition $M_c \leftarrow M_a + M_b$ in the register with rounding to nearest.

(iii) Normalization:
- If $|M_c| = 0$, return 0.
- If $|M_c| \ge \beta$, set $M_c \leftarrow M_c/\beta$ and $e_c \leftarrow e_c + 1$.
- If $|M_c| \in (0, 1)$, repeat $M_c \leftarrow M_c\beta$, $e_c \leftarrow e_c - 1$ until $|M_c| \in [1, \beta)$.

(iv) Check range:
- return NaN if e_c overflows,
- return 0 if e_c underflows.

(v) Round M_c (to nearest) to precision p.

(vi) Set $c \leftarrow M_c \times \beta^{e_c}$.

Example 1.9. Consider the calculation of $c := \mathrm{fl}(a+b)$ with $a = 1.234 \times 10^4$ and $b = 5.678 \times 10^0$ in an FPN system $\mathcal{F} : (10, 4, -7, 8)$.

(i) $b \leftarrow 0.0005678 \times 10^4$; $e_c \leftarrow 4$.

(ii) $m_c \leftarrow 1.2345678$.

(iii) do nothing.

(iv) do nothing.

(v) $m_c \leftarrow 1.235$.

(vi) $c = 1.235 \times 10^4$.

For $b = 5.678 \times 10^{-2}$, $c = a$ would be returned in step (i).

Example 1.10. Consider the calculation of $c := \mathrm{fl}(a+b)$ with $a = 1.000 \times 10^0$ and $b = -9.000 \times 10^{-5}$ in an FPN system $\mathcal{F} : (10, 4, -7, 8)$.

(i) $b \leftarrow -0.0000900 \times 10^0$; $e_c \leftarrow 0$.

(ii) $m_c \leftarrow 0.9999100$.

(iii) $e_c \leftarrow e_c - 1$; $m_c \leftarrow 9.9991000$.

(iv) do nothing.

(v) $m_c \leftarrow 9.999$.

(vi) $c = 9.999 \times 10^{-1}$.

For $b = -9.000 \times 10^{-6}$, $c = a$ would be returned in step (i).

Exercise 1.11. Repeat Example 1.9 with $b = 8.769 \times 10^4$, $b = -5.678 \times 10^0$, and $b = -5.678 \times 10^3$.

Lemma 1.23. For $a, b \in \mathcal{F}$, $a + b \in \mathcal{R}(\mathcal{F})$ implies

$$\mathrm{fl}(a+b) = (a+b)(1+\delta), \qquad |\delta| \le \epsilon_u. \tag{1.15}$$

Proof. The round-off error in step (v) always dominates that in step (ii), which, because of the $2p$ precision, is nonzero only in the case of $e_a - e_b = p + 1$. Then (1.15) follows from Theorem 1.20. □

Definition 1.24 (Multiplication of two FPNs)**.** Express $a, b \in \mathcal{F}$ as $a = M_a \times \beta^{e_a}$ and $b = M_b \times \beta^{e_b}$ where $M_a = \pm m_a$ and $M_b = \pm m_b$. The product $c := \mathrm{fl}(ab) \in \mathcal{F}$ is calculated in a register of precision at least $p + 2$ as follows.

(i) Exponent sum: $e_c \leftarrow e_a + e_b$.

(ii) Perform the multiplication $M_c \leftarrow M_a M_b$ in the register with rounding to nearest.

(iii) Normalization:
- If $|M_c| \ge \beta$, set $M_c \leftarrow M_c/\beta$ and $e_c \leftarrow e_c + 1$.

(iv) Check range:
- return NaN if e_c overflows,
- return 0 if e_c underflows.

(v) Round M_c (to nearest) to precision p.

(vi) Set $c \leftarrow M_c \times \beta^{e_c}$.

Example 1.12. Consider the calculation of $c := \mathrm{fl}(ab)$ with $a = 2.345 \times 10^4$ and $b = 6.789 \times 10^0$ in an FPN system $\mathcal{F} : (10, 4, -7, 8)$.

(i) $e_c \leftarrow 4$.

(ii) $M_c \leftarrow 15.9202$.

(iii) $m_c \leftarrow 1.59202$, $e_c \leftarrow 5$.

(iv) do nothing.

(v) $m_c \leftarrow 1.592$.

(vi) $c = 1.592 \times 10^5$.

Lemma 1.25. For $a, b \in \mathcal{F}$, $|ab| \in \mathcal{R}(\mathcal{F})$ implies

$$\mathrm{fl}(ab) = (ab)(1+\delta), \qquad |\delta| \le \epsilon_u. \tag{1.16}$$

Proof. The error only comes from the round-off in steps (ii) and (v). Then (1.16) follows from Theorem 1.20. □

Definition 1.26 (Division of two FPNs)**.** Express $a, b \in \mathcal{F}$ as $a = M_a \times \beta^{e_a}$ and $b = M_b \times \beta^{e_b}$ where $M_a = \pm m_a$ and $M_b = \pm m_b$. The quotient $c = \mathrm{fl}\left(\frac{a}{b}\right) \in \mathcal{F}$ is calculated in a register of precision at least $2p+1$ as follows.

(i) If $m_b = 0$, return NaN; otherwise set $e_c \leftarrow e_a - e_b$.

(ii) Perform the division $M_c \leftarrow M_a/M_b$ in the register with rounding to nearest.

(iii) Normalization:

- If $|M_c| < 1$, set $M_c \leftarrow M_c\beta$, $e_c \leftarrow e_c - 1$.

(iv) Check range:

- return NaN if e_c overflows,
- return 0 if e_c underflows.

(v) Round M_c (to nearest) to precision p.

(vi) Set $c \leftarrow M_c \times \beta^{e_c}$.

Lemma 1.27. For $a, b \in \mathcal{F}$, $\frac{a}{b} \in \mathcal{R}(\mathcal{F})$ implies

$$\mathrm{fl}\left(\frac{a}{b}\right) = \frac{a}{b}(1+\delta), \qquad |\delta| < \epsilon_u. \tag{1.17}$$

Proof. In the case of $|M_a| = |M_b|$, there is no rounding error in Definition 1.26 and (1.17) clearly holds. Hereafter we denote by M_{c1} and M_{c2} the results of steps (ii) and (v) in Definition 1.26, respectively.

In the case of $|M_a| > |M_b|$, the condition $a, b \in \mathcal{F}$, Definition 1.9, and $|M_a|, |M_b| \in [1, \beta)$ imply

$$\left|\frac{M_a}{M_b}\right| \ge \frac{\beta - \epsilon_M}{\beta - 2\epsilon_M} > 1 + \beta^{-1}\epsilon_M, \tag{1.18}$$

which further implies that the normalization step (iii) in Definition 1.26 is not invoked. By Lemma 1.19, the unit roundoff for the register is $\beta^{-2}\epsilon_u\epsilon_M$. Therefore we have

$$\begin{aligned}
M_{c2} &= M_{c1} + \delta_2, \qquad |\delta_2| \le \epsilon_u \\
&= \frac{M_a}{M_b} + \delta_1 + \delta_2, \qquad |\delta_1| \le \beta^{-2}\epsilon_u\epsilon_M \\
&= \frac{M_a}{M_b}(1+\delta); \\
|\delta| &= \left|\frac{\delta_1 + \delta_2}{M_a/M_b}\right| < \frac{\epsilon_u\left(1 + \beta^{-2}\epsilon_M\right)}{1 + \beta^{-1}\epsilon_M} < \epsilon_u,
\end{aligned}$$

where we have applied (1.18) and the triangular inequality in deriving the first inequality of the last line.

Consider the last case $|M_a| < |M_b|$. It is impossible to have $|M_{c1}| = 1$ in step (ii) because

$$\frac{|M_a|}{|M_b|} \le \frac{\beta - 2\epsilon_M}{\beta - \epsilon_M} = 1 - \frac{\epsilon_M}{\beta - \epsilon_M} < 1 - \beta^{-1}\epsilon_M$$

and the precision of the register is greater than $p+1$. Therefore $|M_{c1}| < 1$ must hold and in Definition 1.26 step (iii) is invoked to yield

$$\begin{aligned}
M_{c1} &= \frac{M_a}{M_b} + \delta_1, \qquad |\delta_1| \le \beta^{-2}\epsilon_u\epsilon_M; \\
M_{c2} &= \beta M_{c1} + \delta_2, \qquad |\delta_2| \le \epsilon_u \\
&= \beta\frac{M_a}{M_b}\left(1 + \frac{\beta\delta_1 + \delta_2}{\beta M_a/M_b}\right),
\end{aligned}$$

where the denominator in the parentheses satisfies

$$\beta\left|\frac{M_a}{M_b}\right| \ge \frac{\beta}{\beta - \epsilon_M} > 1 + \beta^{-1}\epsilon_M.$$

Hence we have

$$|\delta| = \left|\frac{\beta\delta_1 + \delta_2}{\beta M_a/M_b}\right| < \frac{\beta^{-1}\epsilon_u\epsilon_M + \epsilon_u}{1 + \beta^{-1}\epsilon_M} = \epsilon_u. \qquad \square$$

Theorem 1.28 (Model of machine arithmetic)**.** Denote by $\mathcal{F}$ a normalized FPN system with precision p. For each arithmetic operation $\odot = +, -, \times, /$, we have

$$\forall a, b \in \mathcal{F},\ a \odot b \in \mathcal{R}(\mathcal{F}) \ \Rightarrow\ \mathrm{fl}(a \odot b) = (a \odot b)(1+\delta) \tag{1.19}$$

where $|\delta| \le \epsilon_u$ if and only if these binary operations are performed in a register with precision $2p+1$.

Proof. This follows from Lemmas 1.23, 1.25, and 1.27. □

1.2.3 The propagation of rounding errors

Theorem 1.29. If $\forall i = 0, 1, \cdots, n$, $a_i \in \mathcal{F}$, $a_i > 0$, then

$$\mathrm{fl}\left(\sum_{i=0}^{n} a_i\right) = (1 + \delta_n)\sum_{i=0}^{n} a_i, \tag{1.20}$$

where $|\delta_n| < (1+\epsilon_u)^n - 1 \approx n\epsilon_u$.

Proof. Define $s_k := \sum_{i=0}^{k} a_i$,

$$\begin{cases} s_0 & := a_0; \\ s_{k+1} & := s_k + a_{k+1}, \end{cases} \qquad \begin{cases} s_0^* & := a_0; \\ s_{k+1}^* & := \mathrm{fl}(s_k^* + a_{k+1}), \end{cases}$$

$$\delta_k := \frac{s_k^* - s_k}{s_k}, \qquad \epsilon_k := \frac{s_{k+1}^* - (s_k^* + a_{k+1})}{s_k^* + a_{k+1}},$$

and we have

$$\begin{aligned}
\delta_{k+1} &= \frac{s_{k+1}^* - s_{k+1}}{s_{k+1}} = \frac{(s_k^* + a_{k+1})(1+\epsilon_k) - s_{k+1}}{s_{k+1}} \\
&= \frac{(s_k(1+\delta_k) + a_{k+1})(1+\epsilon_k) - s_k - a_{k+1}}{s_{k+1}} \\
&= \frac{(\epsilon_k + \delta_k + \epsilon_k\delta_k)s_k + \epsilon_k a_{k+1}}{s_{k+1}} \\
&= \frac{\epsilon_k s_{k+1} + \delta_k(1+\epsilon_k)s_k}{s_{k+1}} = \epsilon_k + \delta_k(1+\epsilon_k)\frac{s_k}{s_{k+1}}.
\end{aligned}$$

The condition of a_i's being positive implies $s_k < s_{k+1}$, and Theorem 1.20 states $|\epsilon_k| < \epsilon_u$. Hence we have

$$|\delta_{k+1}| < |\epsilon_k| + |\delta_k|(1+\epsilon_u) \le \epsilon_u + |\delta_k|(1+\epsilon_u).$$

An easy induction then shows that

$$\begin{aligned}\forall k \in \mathbb{N},\ |\delta_{k+1}| &< \epsilon_u \sum_{i=0}^{k} (1+\epsilon_u)^i \qquad (1.21)\\ &= \epsilon_u \frac{(1+\epsilon_u)^{k+1}-1}{1+\epsilon_u-1} = (1+\epsilon_u)^{k+1}-1,\end{aligned}$$

where the second step follows from the summation formula of geometric series. The proof is completed by the binomial theorem. □

Exercise 1.13. If we sort the positive numbers $a_i > 0$ according to their magnitudes and carry out the additions in this ascending order, we can minimize the rounding error term δ in Theorem 1.29. Can you give some examples?

Exercise 1.14. Derive $\mathrm{fl}(a_1b_1 + a_2b_2 + a_3b_3)$ for $a_i, b_i \in \mathcal{F}$ and make some observations on the corresponding derivation of $\mathrm{fl}(\sum_i \prod_j a_{i,j})$.

Theorem 1.30. For given $\mu \in \mathbb{R}^+$ and a positive integer $n \le \lfloor \frac{\ln 2}{\mu} \rfloor$, suppose $|\delta_i| \le \mu$ for each $i = 1, 2, \ldots, n$. Then

$$1 - n\mu \le \prod_{i=1}^{n}(1+\delta_i) \le 1 + n\mu + (n\mu)^2, \qquad (1.22)$$

or equivalently, for $I_n := [-\frac{1}{1+n\mu}, 1]$,

$$\exists \theta \in I_n \text{ s.t. } \prod_{i=1}^{n}(1+\delta_i) = 1 + \theta(n\mu + n^2\mu^2). \qquad (1.23)$$

Proof. The condition $|\delta_i| \le \mu$ implies

$$(1-\mu)^n \le \prod_{i=1}^{n}(1+\delta_i) \le (1+\mu)^n.$$

Taylor expansion of $f(\mu) = (1-\mu)^n$ at $\mu = 0$ with Lagrangian remainder yields

$$(1-\mu)^n \ge 1 - n\mu,$$

which implies the first inequality in (1.22). On the other hand, the Taylor series of e^x for $x \in \mathbb{R}^+$ satisfies

$$\begin{aligned} e^x &= 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots \\ &= 1 + x + \frac{x^2}{2!}\left(1 + \frac{x}{3} + \frac{2x^2}{4!} + \cdots\right) \\ &\le 1 + x + \frac{x^2}{2} e^x. \end{aligned}$$

Set $x = n\mu$ in the above inequality, apply the condition $n\mu \le \ln 2$, and we have

$$e^{n\mu} \le 1 + n\mu + (n\mu)^2,$$

which, together with the inequality $(1+\mu)^n \le e^{n\mu}$, yields the second inequality in (1.22).

Finally, (1.22) implies that $\prod_{i=1}^{n}(1+\delta_i)$ is in the range of the continuous function $f(\tau) = 1 + \tau(1+n\mu)n\mu$ on I_n. The rest of the proof follows from the intermediate value theorem. □

1.3 Accuracy and stability

1.3.1 Avoiding catastrophic cancellation

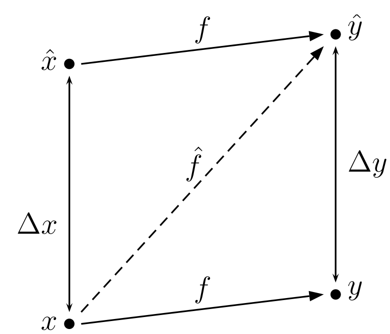
Definition 1.31. Let $\hat{x}$ be an approximation to $x \in \mathbb{R}$. The accuracy of $\hat{x}$ can be measured by its *absolute error*

$$E_{\mathrm{abs}}(\hat{x}) = |\hat{x} - x| \qquad (1.24)$$

and/or its *relative error*

$$E_{\mathrm{rel}}(\hat{x}) = \frac{|\hat{x} - x|}{|x|}. \qquad (1.25)$$

Definition 1.32. For an approximation $\hat{y}$ to $y = f(x)$ computed by $\hat{y} = \hat{f}(x)$, the *forward error* is the relative error of $\hat{y}$ in approximating y and the *backward error* is the smallest relative error in approximating x by an $\hat{x}$ that satisfies $f(\hat{x}) = \hat{f}(x)$, assuming such an $\hat{x}$ exists.



Definition 1.33 (Accuracy)**.** An algorithm $\hat{y} = \hat{f}(x)$ for computing the function $y = f(x)$ is *accurate* if its forward error is small for all x, i.e. $\forall x \in \mathrm{dom}(f)$, $E_{\mathrm{rel}}(\hat{f}(x)) \le c\epsilon_u$ where c is a small constant.

Example 1.15 (Catastrophic cancellation)**.** For two real numbers $x, y \in \mathcal{R}(\mathcal{F})$, Theorems 1.20 and 1.28 imply

$$\begin{aligned}\mathrm{fl}(\mathrm{fl}(x) \odot \mathrm{fl}(y)) &= (\mathrm{fl}(x) \odot \mathrm{fl}(y))(1+\delta_3) \\ &= (x(1+\delta_1) \odot y(1+\delta_2))(1+\delta_3)\end{aligned}$$

where $|\delta_i| \le \epsilon_u$. From Theorems 1.28 and 1.30, we know that *multiplication is accurate*:

$$\begin{aligned}\mathrm{fl}(\mathrm{fl}(x) \times \mathrm{fl}(y)) &= xy(1+\delta_1)(1+\delta_2)(1+\delta_3) \\ &= xy(1 + \theta(3\epsilon_u + 9\epsilon_u^2)),\end{aligned}$$

where $\theta \in [-1, 1]$. Similarly, *division is also accurate*:

$$\begin{aligned}&\mathrm{fl}(\mathrm{fl}(x)/\mathrm{fl}(y)) = \frac{x(1+\delta_1)}{y(1+\delta_2)}(1+\delta_3) \\ =& \frac{x}{y}(1+\delta_1)(1-\delta_2+\delta_2^2-\cdots)(1+\delta_3) \\ \approx& \frac{x}{y}(1+\delta_1)(1-\delta_2)(1+\delta_3).\end{aligned}$$

However, *addition and subtraction might not be accurate*:

$$\begin{aligned}&\mathrm{fl}(\mathrm{fl}(x)+\mathrm{fl}(y)) = (x(1+\delta_1)+y(1+\delta_2))(1+\delta_3)\\ =&(x+y+x\delta_1+y\delta_2)(1+\delta_3)\\ =&(x+y)\left(1+\delta_3+\frac{x\delta_1+y\delta_2}{x+y}+\delta_3\frac{x\delta_1+y\delta_2}{x+y}\right).\end{aligned}$$

In other words, the relative error of addition or subtraction can be arbitrarily large when $x+y\to 0$.

Theorem 1.34 (Loss of most significant digits)**.** Suppose $x,y\in\mathcal{F}$, $x>y>0$, and

$$\beta^{-t}\le 1-\frac{y}{x}\le\beta^{-s}. \tag{1.26}$$

Then the number of most significant digits that are lost in the subtraction $x-y$ is at most t and at least s.

Proof. Rewrite $x=m_x\times\beta^n$ and $y=m_y\times\beta^m$ with $1\le m_x,m_y<\beta$. Definition 1.22 and the condition $x>y$ imply that m_y, the significand of y, is shifted so that y has the same exponent as x before m_x-m_y is performed in the register. Then

$$\begin{aligned}&y=(m_y\times\beta^{m-n})\times\beta^n\\ \Rightarrow\ &x-y=(m_x-m_y\times\beta^{m-n})\times\beta^n\\ \Rightarrow\ &m_{x-y}=m_x\left(1-\frac{m_y\times\beta^m}{m_x\times\beta^n}\right)=m_x\left(1-\frac{y}{x}\right)\\ \Rightarrow\ &\beta^{-t}\le m_{x-y}<\beta^{1-s}.\end{aligned}$$

To normalize m_{x-y} into the interval $[1,\beta)$, it should be multiplied by at least β^s and at most β^t. In other words, m_{x-y} should be shifted to the left for at least s times and at most t times. Therefore the conclusion on the number of lost significant digits follows. □

Rule 1.35. Catastrophic cancellation should be avoided whenever possible.

Example 1.16. Calculate $y=f(x)=x-\sin x$ for $x\to 0$. When x is small, a straightforward calculation would result in a catastrophic cancellation because $x\approx\sin x$. The solution is to use the Taylor series

$$\begin{aligned}x-\sin x&=x-\left(x-\frac{x^3}{3!}+\frac{x^5}{5!}-\frac{x^7}{7!}+\cdots\right)\\ &=\frac{x^3}{3!}-\frac{x^5}{5!}+\frac{x^7}{7!}+\cdots\end{aligned}$$

1.3.2 Backward stability and numerical stability

Definition 1.36 (Backward stability)**.** An algorithm $\hat{f}(x)$ for computing $y=f(x)$ is *backward stable* if its backward error is small for all x, i.e.

$$\begin{aligned}&\forall x\in\mathrm{dom}(f),\ \exists\hat{x}\in\mathrm{dom}(f),\quad\text{s.t.}\\ &\hat{f}(x)=f(\hat{x})\ \Rightarrow\ E_{\mathrm{rel}}(\hat{x})\le c\epsilon_u,\end{aligned}\tag{1.27}$$

where c is a small constant.

Definition 1.37. An algorithm $\hat{f}(x_1,x_2)$ for computing $y=f(x_1,x_2)$ is *backward stable* if

$$\begin{aligned}&\forall(x_1,x_2)\in\mathrm{dom}(f),\ \exists(\hat{x}_1,\hat{x}_2)\in\mathrm{dom}(f)\text{ s.t.}\\ &\hat{f}(x_1,x_2)=f(\hat{x}_1,\hat{x}_2)\ \Rightarrow\ \begin{cases}E_{\mathrm{rel}}(\hat{x}_1)\le c_1\epsilon_u,\\ E_{\mathrm{rel}}(\hat{x}_2)\le c_2\epsilon_u,\end{cases}\end{aligned}\tag{1.28}$$

where c_1, c_2 are two small constants.

Corollary 1.38. For $f(x_1,x_2)=x_1-x_2$, $x_1,x_2\in\mathcal{R}(\mathcal{F})$, the algorithm $\hat{f}(x_1,x_2)=\mathrm{fl}(\mathrm{fl}(x_1)-\mathrm{fl}(x_2))$ is backward stable.

Proof. We have $\hat{f}(x_1,x_2)=(\mathrm{fl}(x_1)-\mathrm{fl}(x_2))(1+\delta_3)$ from Theorem 1.28. Then Theorem 1.20 implies

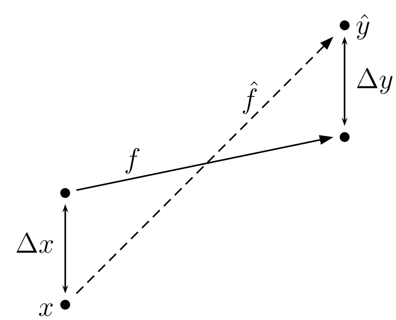
$$\begin{aligned}\hat{f}(x_1,x_2)&=(x_1(1+\delta_1)-x_2(1+\delta_2))(1+\delta_3)\\ &=x_1(1+\delta_1+\delta_3+\delta_1\delta_3)-x_2(1+\delta_2+\delta_3+\delta_2\delta_3).\end{aligned}$$

Take $\hat{x}_1$ and $\hat{x}_2$ to be the two terms in the above line and we have

$$\begin{aligned}E_{\mathrm{rel}}(\hat{x}_1)&=|\delta_1+\delta_3+\delta_1\delta_3|,\\ E_{\mathrm{rel}}(\hat{x}_2)&=|\delta_2+\delta_3+\delta_2\delta_3|.\end{aligned}$$

Then Definition 1.37 completes the proof. □

Example 1.17. For $f(x)=1+x$, $x\in(0,\mathrm{OFL})$, show that the algorithm $\hat{f}(x)=\mathrm{fl}(1.0+\mathrm{fl}(x))$ is not backward stable.



Definition 1.39. An algorithm $\hat{f}(x)$ for computing $y=f(x)$ is *stable* or *numerically stable* iff

$$\forall x\in\mathrm{dom}(f),\ \exists\hat{x}\in\mathrm{dom}(f)\text{ s.t. }\begin{cases}\left|\frac{\hat{f}(x)-f(\hat{x})}{f(\hat{x})}\right|\le c_f\epsilon_u,\\ E_{\mathrm{rel}}(\hat{x})\le c\epsilon_u,\end{cases}\tag{1.29}$$

where c_f, c are two small constants.

Corollary 1.40. If an algorithm is backward stable, then it is numerically stable.

Proof. By Definition 1.36, $f(\hat{x})=\hat{f}(x)$, hence $c_f=0$. The other condition also follows trivially. □

Example 1.18. For $f(x)=1+x$, $x\in(0,\mathrm{OFL})$, show that the algorithm $\hat{f}(x)=\mathrm{fl}(1.0+\mathrm{fl}(x))$ is stable.

1.3.3 Condition numbers: scalar functions

Definition 1.41. The (relative) *condition number of a function $y = f(x)$* is a measure of the relative change in the output for a small change in the input,

$$C_f(x) = \left|\frac{xf'(x)}{f(x)}\right|. \tag{1.30}$$

Definition 1.42. A problem with a low condition number is said to be *well-conditioned*. A problem with a high condition number is said to be *ill-conditioned*.

Example 1.19. Definition 1.41 yields

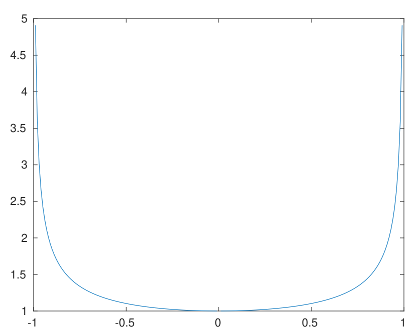
$$E_{\text{rel}}(\hat{y}) \lessapprox C_f E_{\text{rel}}(\hat{x}). \tag{1.31}$$

The approximation mark "$\approx$" refers to the fact that the quadratic term $(\Delta x)^2$ has been ignored. As one way to interpret (1.31) and to understand Definition 1.41, *the computed solution to an ill-conditioned problem may have a large forward error.*

Example 1.20. For the function $f(x) = \arcsin(x)$, its condition number, according to Definition 1.41, is

$$C_f(x) = \left|\frac{xf'(x)}{f(x)}\right| = \frac{x}{\sqrt{1-x^2}\arcsin x}.$$

Hence $C_f(x) \to +\infty$ as $x \to \pm 1$.



Corollary 1.43. Consider solving the equation $f(x) = 0$ near a simple root r, i.e. $f(r) = 0$ and $f'(r) \neq 0$. Suppose we perturb the function f to $F = f + \epsilon g$ where $f, g \in \mathcal{C}^2$, $g(r) \neq 0$, and $|\epsilon g'(r)| \ll |f'(r)|$. Then the root of F is $r + h$ where

$$h \approx -\epsilon \frac{g(r)}{f'(r)}. \tag{1.32}$$

Proof. Suppose $r + h$ is the new root, i.e. $F(r + h) = 0$, or,

$$f(r+h) + \epsilon g(r+h) = 0.$$

Taylor's expansion of $F(r + h)$ yields

$$f(r) + hf'(r) + \epsilon[g(r) + hg'(r)] = O(h^2)$$

and we have

$$h \approx -\epsilon \frac{g(r)}{f'(r) + \epsilon g'(r)} \approx -\epsilon \frac{g(r)}{f'(r)}.$$ □

Example 1.21 (Wilkinson)**.** Define

$$f(x) := \prod_{k=1}^{p}(x-k),$$
$$g(x) := x^p.$$

How is the root $x = p$ affected by perturbing f to $f + \epsilon g$?

By Corollary 1.43, the answer is

$$h \approx -\epsilon \frac{g(p)}{f'(p)} = -\epsilon \frac{p^p}{(p-1)!}.$$

For $p = 20, 30, 40$, the value of $\frac{p^p}{(p-1)!}$ is about 8.6×10^8, 2.3×10^{13}, 5.9×10^{17}, respectively. Hence a small change of the coefficient in the monomial x^p would cause a large change of the root. Consequently, the problem of root finding for polynomials with very high degrees is hopeless.

1.3.4 Condition numbers: vector functions

Definition 1.44. The *condition number of a vector function* $\mathbf{f} : \mathbb{R}^m \to \mathbb{R}^n$ is

$$\text{cond}_{\mathbf{f}}(\mathbf{x}) = \frac{\|\mathbf{x}\| \, \|\nabla \mathbf{f}\|}{\|\mathbf{f}(\mathbf{x})\|}, \tag{1.33}$$

where $\|\cdot\|$ denotes a Euclidean norm such as the 1-, 2-, and ∞-norms.

Example 1.22. In solving the linear system $A\mathbf{u} = \mathbf{b}$, the algorithm can be viewed as taking the input $\mathbf{b}$ and returning the output $A^{-1}\mathbf{b}$, i.e. $\mathbf{f}(\mathbf{b}) = A^{-1}\mathbf{b}$. Clearly $\nabla \mathbf{f} = A^{-1}$. Definition 1.44 yields

$$\text{cond}_{\mathbf{f}}(\mathbf{x}) = \frac{\|\mathbf{b}\| \left\|A^{-1}\right\|}{\|\mathbf{u}\|} = \frac{\|A\mathbf{u}\| \left\|A^{-1}\right\|}{\|\mathbf{u}\|}.$$

In practice the input $\mathbf{b}$ can take any value, hence we have

$$\max \text{cond}_{\mathbf{f}}(\mathbf{x}) = \max \frac{\|A\mathbf{u}\| \left\|A^{-1}\right\|}{\|\mathbf{u}\|} = \|A\| \left\|A^{-1}\right\|,$$

where the last expression is the condition number of A defined in linear algebra and we have used the common definition

$$\|A\| := \max_{\|\mathbf{u}\| \neq 0} \frac{\|A\mathbf{u}\|}{\|\mathbf{u}\|}. \tag{1.34}$$

The above discussion explains why the condition number of a matrix A is usually defined as

$$\text{cond}\, A = \|A\| \left\|A^{-1}\right\|. \tag{1.35}$$

Definition 1.45. The *componentwise condition number* of a vector function $\mathbf{f} : \mathbb{R}^m \to \mathbb{R}^n$ is

$$\text{cond}_{\mathbf{f}}(\mathbf{x}) = \|A(\mathbf{x})\|, \tag{1.36}$$

where the matrix $A(\mathbf{x}) = [a_{ij}(\mathbf{x})]$ and each component is

$$a_{ij}(\mathbf{x}) = \left|\frac{x_j \frac{\partial f_i}{\partial x_j}}{f_i(\mathbf{x})}\right|. \tag{1.37}$$

Example 1.23. For the vector function

$$\mathbf{f}(\mathbf{x}) := \begin{bmatrix} \frac{1}{x_1} + \frac{1}{x_2} \\ \frac{1}{x_1} - \frac{1}{x_2} \end{bmatrix},$$

its Jacobian matrix is

$$\nabla \mathbf{f} = -\frac{1}{x_1^2 x_2^2} \begin{bmatrix} x_2^2 & x_1^2 \\ x_2^2 & -x_1^2 \end{bmatrix}.$$

The condition number based on Definition 1.45 clearly captures the fact that $x_1 \pm x_2 \approx 0$ leads to ill-conditioning,

$$C_c = \begin{bmatrix} \left|\frac{x_2}{x_1+x_2}\right| & \left|\frac{x_1}{x_1+x_2}\right| \\ \left|\frac{x_2}{x_1-x_2}\right| & \left|\frac{x_1}{x_1-x_2}\right| \end{bmatrix},$$

while that based on 1-norm of Definition 1.44 fails to capture the ill-conditioning,

$$C_1 = \frac{\|\mathbf{x}\|_1 \|\nabla \mathbf{f}\|_1}{\|\mathbf{f}\|_1} = \frac{|x_1| + |x_2|}{|x_1 x_2|} \frac{2\max(x_1^2, x_2^2)}{|x_1 + x_2| + |x_1 - x_2|},$$

in that the condition $x_1 \pm x_2 \approx 0$ yields $C_1 \approx 2$. Note that we have used the well-known formula

$$\forall A \in \mathbb{R}^{n\times n}, \qquad \|A\|_1 = \max_j \sum_i |a_{ij}|.$$

Definition 1.46. The *Hilbert matrix* $H_n \in \mathbb{R}^{n\times n}$ is

$$h_{i,j} = \frac{1}{i+j-1}. \tag{1.38}$$

Definition 1.47. The *Vandermonde matrix* $V_n \in \mathbb{R}^{n\times n}$ is

$$v_{i,j} = t_j^{i-1}, \tag{1.39}$$

where $t_1, t_2, \ldots, t_n$ are parameters.

1.3.5 Condition numbers: algorithms

Definition 1.48. Consider approximating a function $\mathbf{f} : \mathbb{R}^m \to \mathbb{R}^n$ with an algorithm $\mathbf{f}_A : \mathcal{F}^m \to \mathcal{F}^n$. Assume

$$\forall \mathbf{x} \in \mathcal{F}^m,\ \exists \mathbf{x}_A \in \mathbb{R}^m \text{ s.t. } \mathbf{f}_A(\mathbf{x}) = \mathbf{f}(\mathbf{x}_A), \tag{1.40}$$

the *condition number of the algorithm* $\mathbf{f}_A$ is defined as

$$\mathrm{cond}_A(\mathbf{x}) = \frac{1}{\epsilon_u} \inf_{\{\mathbf{x}_A\}} \frac{\|\mathbf{x}_A - \mathbf{x}\|}{\|\mathbf{x}\|}. \tag{1.41}$$

Example 1.24. Consider an algorithm A for calculating $y = \ln x$. Suppose that, for any positive number x, this program produces a y_A satisfying $y_A = (1+\delta)\ln x$ where $|\delta| \le 5\epsilon_u$. What is the condition number of the algorithm?

Theorem 1.49. Suppose a smooth function $f : \mathbb{R} \to \mathbb{R}$ is approximated by an algorithm $A : \mathcal{F} \to \mathcal{F}$, producing $f_A(x) = f(x)(1+\delta(x))$ where $|\delta(x)| \le \varphi(x)\epsilon_u$. If $\mathrm{cond}_f(x)$ is bounded, then $\forall x \in \mathcal{F}$,

$$\mathrm{cond}_A(x) \le \frac{\varphi(x)}{\mathrm{cond}_f(x)}. \tag{1.42}$$

Proof. Assume $\forall x$, $\exists x_A$ such that $f(x_A) = f_A(x)$. Write $x_A = x(1+\epsilon_A)$ and we have

$$\begin{aligned} f(x)(1+\delta) &= f(x_A) = f(x(1+\epsilon_A)) = f(x + x\epsilon_A) \\ &= f(x) + x\epsilon_A f'(x) + O(\epsilon_A^2). \end{aligned}$$

Neglecting the quadratic term yields

$$\begin{aligned} & x\epsilon_A f'(x) = f(x)\delta \\ \Rightarrow & \left|\frac{x_A - x}{x}\right| = |\epsilon_A| = \left|\frac{f(x)}{xf'(x)}\right| |\delta(x)|. \end{aligned}$$

Dividing both sides by ϵ_u yields

$$\frac{1}{\epsilon_u}\left|\frac{x_A - x}{x}\right| = \frac{\delta(x)}{\epsilon_u \mathrm{cond}_f(x)}.$$

Take inf with respect to all x_A's, take sup with respect to x, and we have (1.42). $\square$

Example 1.25. Assume that $\sin x$ and $\cos x$ are computed with relative error within machine roundoff (this can be satisfied easily by truncating the Taylor series). Apply Theorem 1.49 to analyze the condition of the algorithm

$$f_A = \mathrm{fl}\left[\frac{\mathrm{fl}\big(1 - \mathrm{fl}(\cos x)\big)}{\mathrm{fl}(\sin x)}\right] \tag{1.43}$$

that computes $f(x) = \frac{1-\cos x}{\sin x}$ for $x \in (0, \pi/2)$.

Exercise 1.26. Repeat Example 1.25 for $f(x) = \frac{\sin x}{1+\cos x}$ on the same interval.

1.3.6 Overall error of a computer solution

Theorem 1.50. Consider using normalized FPN arithmetics to solve a math problem

$$\mathbf{f} : \mathbb{R}^m \to \mathbb{R}^n, \qquad \mathbf{y} = \mathbf{f}(\mathbf{x}). \tag{1.44}$$

Denote the computer input and output as

$$\mathbf{x}^* \approx \mathbf{x}, \qquad \mathbf{y}_A^* = \mathbf{f}_A(\mathbf{x}^*), \tag{1.45}$$

where $\mathbf{f}_A$ is the algorithm that approximates $\mathbf{f}$. The relative error of approximating $\mathbf{y}$ with $\mathbf{y}_A^*$ can be bounded as

$$E_{\mathrm{rel}}(\mathbf{y}_A^*) \lessapprox E_{\mathrm{rel}}(\mathbf{x}^*)\mathrm{cond}_{\mathbf{f}}(\mathbf{x}) + \epsilon_u \mathrm{cond}_{\mathbf{f}}(\mathbf{x}^*)\,\mathrm{cond}_A(\mathbf{x}^*), \tag{1.46}$$

where the relative error is defined in (1.25).

Proof. By the triangle inequality, we have

$$\begin{aligned} \frac{\|\mathbf{y}_A^* - \mathbf{y}\|}{\|\mathbf{y}\|} &= \frac{\|\mathbf{f}_A(\mathbf{x}^*) - \mathbf{f}(\mathbf{x})\|}{\|\mathbf{f}(\mathbf{x})\|} \\ &\le \frac{\|\mathbf{f}(\mathbf{x}^*) - \mathbf{f}(\mathbf{x})\|}{\|\mathbf{f}(\mathbf{x})\|} + \frac{\|\mathbf{f}_A(\mathbf{x}^*) - \mathbf{f}(\mathbf{x}^*)\|}{\|\mathbf{f}(\mathbf{x})\|}. \end{aligned}$$

By (1.31), the first term is

$$\begin{aligned} \frac{\|\mathbf{f}(\mathbf{x}^*) - \mathbf{f}(\mathbf{x})\|}{\|\mathbf{f}(\mathbf{x})\|} &\lessapprox \mathrm{cond}_{\mathbf{f}}(\mathbf{x}) \frac{\|\mathbf{x}^* - \mathbf{x}\|}{\|\mathbf{x}\|} \\ &= E_{\mathrm{rel}}(\mathbf{x}^*)\mathrm{cond}_{\mathbf{f}}(\mathbf{x}). \end{aligned}$$

By (1.31) and Definition 1.48, the second term is

$$\begin{aligned}\frac{\|\mathbf{f}_A(\mathbf{x}^*)-\mathbf{f}(\mathbf{x}^*)\|}{\|\mathbf{f}(\mathbf{x})\|} &= \frac{\|\mathbf{f}(\mathbf{x}^*_A)-\mathbf{f}(\mathbf{x}^*)\|}{\|\mathbf{f}(\mathbf{x})\|} \approx \frac{\|\mathbf{f}(\mathbf{x}^*_A)-\mathbf{f}(\mathbf{x}^*)\|}{\|\mathbf{f}(\mathbf{x}^*)\|} \\ &\le \operatorname{cond}_{\mathbf{f}}(\mathbf{x}^*)\frac{\|\mathbf{x}^*_A-\mathbf{x}^*\|}{\|\mathbf{x}^*\|} \\ &= \epsilon_u \operatorname{cond}_A(\mathbf{x}^*)\operatorname{cond}_{\mathbf{f}}(\mathbf{x}^*),\end{aligned}$$

where the last step follows from the fact that we only consider the $\mathbf{x}^*_A$ that is the least dangerous. □

1.4 Problems

1.4.1 Theoretical questions

I. Convert the decimal integer 477 to a normalized FPN with $\beta = 2$.

II. Convert the decimal fraction 3/5 to a normalized FPN with $\beta = 2$.

III. Let $x = \beta^e$, $e \in \mathbb{Z}$, $L < e < U$ be a normalized FPN in $\mathbb{F}$ and $x_L, x_R \in \mathbb{F}$ the two normalized FPNs adjacent to x such that $x_L < x < x_R$. Prove $x_R - x = \beta(x - x_L)$.

IV. By reusing your result of II, find out the two normalized FPNs adjacent to $x = 3/5$ under the IEEE 754 single-precision protocol. What is $\mathrm{fl}(x)$ and the relative roundoff error?

V. If the IEEE 754 single-precision protocol did not round off numbers to the nearest, but simply dropped excess bits, what would the unit roundoff be?

VI. How many bits of precision are lost in the subtraction $1 - \cos x$ when $x = \frac{1}{4}$?

VII. Suggest at least two ways to compute $1 - \cos x$ to avoid catastrophic cancellation caused by subtraction.

VIII. What are the condition numbers of the following functions? Where are they large?

- $(x-1)^\alpha$,
- $\ln x$,
- e^x,
- $\arccos x$.

IX. Consider the function $f(x) = 1 - e^{-x}$ for $x \in [0,1]$.

- Show that $\operatorname{cond}_f(x) \le 1$ for $x \in [0,1]$.
- Let A be the algorithm that evaluates $f(x)$ for the machine number $x \in \mathbb{F}$. Assume that the exponential function is computed with relative error within machine roundoff. Estimate $\operatorname{cond}_A(x)$ for $x \in [0,1]$.
- Plot $\operatorname{cond}_f(x)$ and $\operatorname{cond}_A(x)$ as a function of x on $[0,1]$. Discuss your results.

X. The math problem of root finding for a polynomial

$$q(x) = \sum_{i=0}^{n} a_i x^i, \qquad a_n = 1, a_0 \ne 0, a_i \in \mathbb{R} \tag{1.47}$$

can be considered as a vector function $f : \mathbb{R}^n \to \mathbb{C}$:

$$r = f(a_0, a_1, \ldots, a_{n-1}).$$

Derive the componentwise condition number of f based on the 1-norm. For the Wilkinson example, compute your condition number, and compare your result with that in the Wilkinson Example. What does the comparison tell you?

1.4.2 Programming assignments

A. Print values of the functions in (1.48) at 101 equally spaced points covering the interval $[0.99, 1.01]$. Calculate each function in a straightforward way without rearranging or factoring. Note that the three functions are theoretically the same, but the computed values might be very different. Plot these functions near 1.0 using a magnified scale for the function values to see the variations involved. Discuss what you see. Which one is the most accurate? Why?

B. Consider a normalized FPN system $\mathbb{F}$ with the characterization $\beta = 2, p = 3, L = -1, U = +1$.

- compute $\mathrm{UFL}(\mathbb{F})$ and $\mathrm{OFL}(\mathbb{F})$ and output them as decimal numbers;
- enumerate all numbers in $\mathbb{F}$ and verify the corollary on the cardinality of $\mathbb{F}$ in the summary handout;
- plot $\mathbb{F}$ on the real axis;
- enumerate all the subnormal numbers of $\mathbb{F}$;
- plot the *extended* $\mathbb{F}$ on the real axis.

$$f(x) = x^8 - 8x^7 + 28x^6 - 56x^5 + 70x^4 - 56x^3 + 28x^2 - 8x + 1 \tag{1.48a}$$

$$g(x) = (((((((x-8)x+28)x-56)x+70)x-56)x+28)x-8)x+1 \tag{1.48b}$$

$$h(x) = (x-1)^8 \tag{1.48c}$$