Numerical Methods

Derek Li

Contents

1 Scientific Computing

1.1 Approximations in Scientific Computation

1.1.1 Absolute Error and Relative Error

Let A be approximate value, T be true value. Absolute error and relative error are defined as follows:

$$\text{Absolute Error} = A - T,$$
$$\text{Relative Error} = \frac{A-T}{T} \text{ assuming } T \neq 0.$$

If numbers written in scientific notation agree to p significant digits, then the magnitude of the relative error is about 10^{-p} (within a factor of 10).

Example 1.1. $A = 5.46729 \times 10^{-12}, T = 5.46417 \times 10^{-12}$. Thus, $A - T = 0.00312 \times 10^{-12}$ and $\frac{A-T}{T} = \frac{3.12\times10^{-3}}{5.46417}$.

Example 1.2. $A = 1.00596 \times 10^{-10}, T = 0.99452 \times 10^{-10}$. Thus, $A - T = 0.01144 \times 10^{-10}$ and $\frac{A-T}{T} = \frac{1.144\times10^{-2}}{0.99452}$. A and T agree to 2 significant digits.

1.1.2 Data Error and Computational Error

The difference between exact function values due to error in the input and thus can be viewed as data error.

The difference between the exact and approximate functions for the same input and thus can be considered computational error.

1.1.3 Truncation Error

Truncation error is the difference between the true result (for the actual input) and the result that would be produced by a given algorithm using exact arithmetic. It is due to approximations such as truncating an infinite series, replacing derivatives by finite differences, or terminating an iterative sequence before convergence.

Example 1.3. $f(x) = \sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \cdots, \widehat{f}(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!}$. The part $-\frac{x^7}{7!} + \cdots$ is the truncation error.

1.1.4 Forward Error and Backward Error

Suppose we want to compute the value of a function, $y = f(x)$, but we obtain instead an approximate value $\widehat{y}$. The discrepancy between the computed and true values, $\Delta y = \widehat{y} - y$, is called the forward error.

The quantity $\Delta x = \widehat{x} - x$, where $f(\widehat{x}) = \widehat{y}$, is called the backward error.

Example 1.4. As an approximation to $y = \sqrt{2}$, the value $\widehat{y} = 1.4$ has a forward error

$$\Delta y = \widehat{y} - y = 1.4 - \sqrt{2} \approx -0.0142$$

or a relative forward error -1.004×10^{-2}. We find that $\sqrt{1.96} = 1.4$, so the backward error is

$$\Delta x = \widehat{x} - x = 1.96 - 2 = -0.04$$

or the relative backward error -2×10^{-2}.

1.1.5 Conditioning

A problem is well-conditioned if a small change to the input produces a small change to the output; ill conditioned if there are some examples for which a small change in the input produces a large change in output.

Consider relative forward error $\frac{\widehat{y}-y}{y} = \frac{f(\widehat{x})-f(x)}{f(x)}$. Since

$$f(\widehat{x}) - f(x) = f'(\widetilde{x})(\widehat{x} - x)$$

for some $\widetilde{x}$ between x and $\widehat{x}$ provided $f'(x)$ exists and is continuous between x and $\widehat{x}$, then

$$\frac{\widehat{y} - y}{y} = \frac{xf'(\widetilde{x})}{f(x)} \frac{\widehat{x} - x}{x} \approx \frac{xf'(x)}{f(x)} \frac{\widehat{x} - x}{x}.$$

$\frac{xf'(x)}{f(x)}$ is called condition number.

Example 1.5. The conditioning of $f(x) = \sqrt{x}, x \geqslant 0$.

Solution. We have $f'(x) = \frac{1}{2\sqrt{x}}$ and thus the condition number is

$$\frac{x}{2\sqrt{x}\sqrt{x}} = \frac{1}{2}.$$

Hence, $f(x)$ is well-conditioned.

Example 1.6. The conditioning of $f(x) = e^x$.

Solution. The condition number is x. Thus, e^x overflows or underflows if $|x|$ is large.

Example 1.7. The conditioning of $f(x) = \sin x$.

Solution. The condition number is $\frac{x \cos x}{\sin x}$.

(1) If $x \approx \pm\pi, \pm 2\pi, \pm 3\pi, \cdots, \sin x$ overflows or underflows. Nevertheless, x could be 0 because

$$\left.\frac{\sin(x)}{x}\right|_{x=0} = \left.\frac{x - \frac{x^3}{3!} + \frac{x^5}{5!} - \cdots}{x}\right|_{x=0} = \left. 1 - \frac{x^2}{3!} + \frac{x^4}{5!} - \cdots \right|_{x=0} = 1.$$

(2) If $|x|$ is big and $\cos x \not\approx 0, \sin x$ overflows or underflows.

1.1.6 Stability of Algorithms

An algorithm is stable if small changes to the input result in small changes to the output. Note that stability of an algorithm does not by itself guarantee that the computed result is accurate.

1.2 Computer Arithmetic

1.2.1 Floating-Point Numbers

In a digital computer, the real number system $\mathbb{R}$ of mathematics is represented by a floating-point number system.

Formally, a floating-point number system $\mathbb{F}$ is characterized by four integers: β (base), p (precision), $[L, U]$ (exponent range). Any floating-point number $x \in \mathbb{F}$ has the form

$$x = \pm d_1.d_2 d_3 \cdots d_p \times \beta^n,$$

where $0 \leqslant d_i < \beta, L \leqslant n \leqslant U$.

1.2.2 IEEE Floating-Point Standard

System	β	p	L	U
Single-Precision	2	24	-126	127
Double-Precision	2	53	-1022	1023

1.2.3 Normalization

A floating-point system is said to be normalized if the leading digit $d_1 \neq 0$ unless the number represented is zero.

1.2.4 Properties of Floating-Point Systems

There is a smallest positive normalized floating-point number, underflow limit

$$\text{UFL} = \underbrace{1.00\cdots0}_{p \text{ digits}} \times \beta^L.$$

There is a largest floating-point number, overflow limit

$$\text{OFL} = \underbrace{d.dd\cdots d}_{p \text{ digits}} \times \beta^U,$$

where $d = \beta - 1$ and thus

$$\text{OFL} = (\beta - \beta^{1-p})\beta^U = (1 - \beta^{-p})\beta^{U+1}.$$

1.2.5 Rounding

One of the commonly used rounding rules is round-to-nearest which is the default rounding rule in IEEE standard system.

Note 1. Most of the time, there is a unique closest p-digit number.

Note 2. In case of times, we round to the number with an even least significant digit (round to even).

Note 3. In binary, in case of times, we round to the number which has a 0 in its least significant digits.

Example 1.8. Consider a system with $\beta = 10, p = 3, L = -10, U = 10$.

$$1.54 \times 10^1 + 2.56 \times 10^{-1} = 1.5656 \times 10^1 \approx 1.57 \times 10^1.$$

1.2.6 Machine Epsilon

Define machine epsilon

$$\varepsilon_{\text{mach}} = \beta^{1-p}.$$

There is same number of floating-point numbers in each interval $[\beta^n, \beta^{n+1})$, and numbers are evenly spaced with distance $\beta^n \cdot \varepsilon_{\text{mach}}$.

The machine epsilon is important because it bounds the relative error in representing any nonzero real number x within the normalized range of a floating-point system:

$$|\text{fl}(x) - x| \leqslant \frac{1}{2}\beta^n \varepsilon_{\text{mach}} \Rightarrow \left|\frac{\text{fl}(x) - x}{x}\right| \leqslant \frac{\frac{1}{2}\beta^n \varepsilon_{\text{mach}}}{\beta^n} = \frac{1}{2}\varepsilon_{\text{mach}}.$$

In IEEE, $\text{fl}(a \text{ op } b)$ is closest flouting-point number to a op b (round to nearest and no overflows or underflows), where op means basic operations: $+, -, \times, /, \sqrt{\cdots}$. We also have

$$\left|\frac{\text{fl}(a \text{ op } b) - (a \text{ op } b)}{a \text{ op } b}\right| \leqslant \frac{1}{2}\varepsilon_{\text{mach}}.$$

1.2.7 Subnormals and Gradual Underflow

There is a noticeable gap around zero in floating-point system because of normalization. Subnormal numbers have $d_1 = 0$, which can fill in the gap, but the inequality

$$\left|\frac{\text{fl}(x) - x}{x}\right| \leqslant \frac{1}{2}\varepsilon_{\text{mach}}$$

might not hold.

Such an augmented floating-point system is sometimes said to exhibit gradual underflow, since it extends the lower range of magnitudes representable rather than underflowing to zero as soon as the minimum exponent value would otherwise be exceeded.

Example 1.9 (Underflow to a Subnormal)**.** Consider a system with $\beta = 10, p = 3, L = -10, U = 10$.

$1.01 \times 10^{-5} \times 2.02 \times 10^{-6} = 2.0402 \times 10^{-11} = 0.20402 \times 10^{-10} \approx 0.20 \times 10^{-10}$.

Example 1.10 (Underflow to Zero)**.**

$1.01 \times 10^{-6} \times 2.02 \times 10^{-7} = 2.0402 \times 10^{-13} = 0.0020402 \times 10^{-10} \approx 0.00 \times 10^{-10}$.

1.2.8 Exceptional Values

The IEEE floating-point standard provides two additional special values to indicate exceptional situations:

- Inf, which stands for infinity, results from dividing a finite number by zero or Inf + Inf.
- NaN, which stands for not a number, results from an undefined or indeterminate operation such as $\frac{0}{0}, 0 \times \text{Inf}, \frac{\text{Inf}}{\text{Inf}}$, or Inf − Inf.

Example 1.11 (Overflow)**.**

$$3.56 \times 10^5 \times 5.41 \times 10^5 = 19.2596 \times 10^{10} \rightarrow \text{Inf}.$$