Numerical Analysis

Dhyan Laad
2024ADPS0875G

Contents

1 Error Analysis

1.1 Floating Point Forms

A real number may potentially have an infinite decimal expansion, but computers are limited by hardware, and as such store numbers with a terminating approximation.

Definition 1.1. Given a real number x with digits $d_1, d_2, \dots$, the *n-digit, base β floating point form*, or *n-β floating point form* is

$$(-1)^s \times (0.d_1d_2\dots d_n)_\beta \times \beta^e$$

where $s \in \{0,1\}$ is the *sign*, e is the *exponent*, and the β-fraction

$$(0.d_1d_2\dots d_n)_\beta = \frac{d_1}{\beta^1} + \frac{d_2}{\beta^2} + \cdots + \frac{d_n}{\beta^n}$$

is called the *mantissa*. In the case that $d_1 \neq 0$, the representation is called the *normalized floating point form*.

For a fixed value of β and n as defined above, the notation $\mathrm{fl}(x)$ is used to denote the n-β floating point representation of x. Furthermore, for all computing systems, there are bounds on the values that the exponent e can take. This leads to the concepts of underflow and overflow.

Definition 1.2. Let a real number x have a floating point form with exponent e. For a computing system with exponential range (m, M) where m and M are integers,

(a) if $e > M$, then the system is said to *overflow*, and the result of the computation is denoted with a signed infinity: $\pm\infty$, and

(b) if $e < m$, then the system is said to *underflow*, and the result of the computation is simply 0.

There are two ways to determine the mantissa of the floating point representation of a real number with more than n digits: chopping and rounding. The chopped mantissa of the floating point representation of $x = 0.d_1d_2\dots d_nd_{n+1}\dots$ would simply be $(0.d_1d_2\dots d_n)$, while the rounded mantissa would be

$$\begin{cases} (0.d_1d_2\dots d_n) & d_{n+1} \in [0, \beta/2), \\ (0.d_1d_2\dots(d_n+1)) & d_{n+1} \in [\beta/2, \beta]. \end{cases}$$

1.2 Errors

The error of a floating point representation is a quantitification of how far removed it is from its true value.

Definition 1.3. Let $x \in \mathbb{R}$. The *absolute error* of its floating point representation is

$$x - \mathrm{fl}(x).$$

Note that since $\mathrm{fl}(x) \leq x$ for all $x \in \mathbb{R}$, the absolute error is always a positive quantity. Absolute error is the simplest quantitification but not the most useful, motivating a definition for relative error.

Definition 1.4. The ratio of the absolute error to the true value of a real number x is called its *relative error*. It is customarily denoted with ε:

$$\varepsilon = \frac{x - \mathrm{fl}(x)}{x}.$$

Another quantitification of how removed an approximation is from its true value is captured in the approximation's significant figures or significant digits.

Definition 1.5. Let x be a real number and x^* be an approximation of it. Then if

$$|x - x^*| \leq \frac{1}{2}\beta^{s-r+1}$$

where s is the largest integer such that $\beta^s \leq |x|$, then x^* is said to approximate x to r *significant figures* in β.

Theorem 1.6. *Let $\mathrm{fl}(x)$ be the n-β floating point representation for $x \in \mathbb{R}$, and set*

$$\varepsilon = \frac{x - \mathrm{fl}(x)}{x}.$$

Then,

(a) $\varepsilon \leq \beta^{-n+1}$ *for chopped systems, and*

(b) $\varepsilon \leq \frac{1}{2}\beta^{-n+1}$ *for rounded systems.*

Proof. Let x be a nonzero real number represented as

$$x = m \cdot \beta^e = (-1)^s \cdot (0.d_1 d_2 \dots d_n d_{n+1} \dots)\beta^e$$

where $d_1 \neq 0$. The smallest possible magnitude for the mantissa is $0.100\dots$ (in base β). Therefore, the bounds on m are

$$\frac{1}{\beta} \leq |m| < 1.$$

Since the floating point representation only stores n digits, the last digit stored is d_n, which is in the β^{-n} position relative to the decimal point.

Now, a chopped system truncates everything after d_n, and the absolute error would be given by

$$|x - \mathrm{fl}(x)| = 0.\underbrace{00\dots0}_{n \text{ zeros}} d_n d_{n+1} \dots \times \beta^e < \beta^{-n} \times \beta^e = \beta^{e-n}.$$

Now consider the relative error.

$$|\varepsilon| = \left|\frac{x - \mathrm{fl}(x)}{x}\right| < \frac{\beta^{e-n}}{|m \times \beta^e|} = \frac{\beta^{-n}}{|m|}.$$

To find the upper bound, we must minimize the denominator, whose minimum value we previously determined to be $1/\beta$, which yields

$$|\varepsilon| < \frac{\beta^{-n}}{1/\beta} \Rightarrow \varepsilon \le \beta^{-n+1}. \tag{a}$$

In a rounded system, $\text{fl}(x)$ is the number with n digits closest to x. The quantity analogous to a "least count" would be β^{e-n}. When rounding, the error cannot exceed half of this value

$$|x - \text{fl}(x)| \le \frac{1}{2}\beta^{-n} \times \beta^{e} = \frac{1}{2}\beta^{e-n}.$$

Dividing by x yields

$$|\varepsilon| = \frac{|x - \text{fl}(x)|}{|x|} \le \frac{1}{2} \cdot \frac{\beta^{-n}}{|m|}.$$

Once more, the error is maximized at $|m| = 1/\beta$. Therefore,

$$\varepsilon \le \frac{1}{2}\beta^{-n+1}. \tag{b}$$

□

Propogation of Errors

When performing the arithmetic operations with approximate quantities, it is important to study the errors in the sum, difference, product, and quotient. Let $x = x^* + \varepsilon$ and $y = y^* + \eta$, where x and y are true real values, and x^* and y^* are their approximations. Let r_n denote the relative error in a quantity n. Then,

$$r_{xy} = \frac{xy - x^*y^*}{xy} = \frac{xy - (x-\varepsilon)(y-\eta)}{xy} = \frac{\varepsilon}{x} + \frac{\eta}{y} - \frac{\varepsilon\eta}{xy} \approx r_x + r_y,$$
$$r_{x/y} = \frac{x(1+r_x)}{y(1+r_y)} = \frac{1+r_x}{1+r_y} - 1 \approx (1+r_x)(1-r_y) - 1 \approx r_x - r_y.$$

These approximations hold for $|r_x|, |r_y| \ll 1$. Errors in addition in subtraction are given by

$$r_{x\pm y} = r_x\left(\frac{x}{x\pm y}\right) \pm r_y\left(\frac{y}{x \pm y}\right).$$

In general, errors propogated through multiplication and division do not propogate rapidly. However, the propogated error in addition and subtraction can be much larger than the relative errors in either x and y. This is known as a loss of significance or catastrophic cancellation.

Example 1.7. Consider the function

$$f : x \mapsto x(\sqrt{x+1} - \sqrt{x}).$$

Evaluate $f(10^5)$ using 6-digit floating point arithmetic with rounding.

Solution. Firstly, compute the floating point forms of the square roots:

$$\begin{aligned}\mathrm{fl}(\sqrt{100,001}) &= 3.16229 \times 10^2\\ \mathrm{fl}(\sqrt{100,000}) &= 3.16228 \times 10^2.\end{aligned}$$

Taking the difference would yield 0.00001×10^2. Starting wih 6 significant digits, we are down to 1, which is a huge loss in precision. The final result would be

$$f(100,000) \approx (1.00000 \times 10^5) \times (1.00000 \times 10^{-3}) = 1.00000 \times 10^2 = 100.$$

However, the true value of $f(10^5)$ is around 158.113488. To avoid the huge loss in precision, we first manipulate the function to remove the subtraction of two nearly equal quantities:

$$f(x) = \frac{x}{\sqrt{x+1} + \sqrt{x}}$$

Performing 6-digit floating point arithmetic on the manipulated function gives 158.114, which is much closer to the true value. ■

We now consider the propogation of errors through function evaluation. Let $x = x^* + \varepsilon$ where $x \in \mathbb{R}$ and x^* is an approximation correct to n significant figures. Let $f : X \to \mathbb{R}$ be a smooth function for some subset X of $\mathbb{R}$. Using its Taylor series, we have

$$f(x^* + \varepsilon) = f(x^*) + \varepsilon f'(x^*) + \frac{\varepsilon^2}{2} f''(\alpha)$$

for some $\alpha \in [x, x^*]$. The relative error is given by

$$r_{f(x)} = \frac{f(x) - f(x^*)}{f(x)} = \frac{\varepsilon f'(x^*) + \varepsilon^2 f''(\alpha)/2}{f(x)} \approx \varepsilon \frac{f'(x^*)}{f(x)}.$$

Assuming that $f(x)$ and $f(x^*)$ are relatively close,

$$r_{f(x)} \approx \frac{\varepsilon}{x} \cdot \frac{x f'(x^*)}{f(x^*)} \Rightarrow \left| r_{f(x)} \right| \approx |r_x| \left| \frac{x^* f'(x^*)}{f(x^*)} \right|.$$

We also denote

$$\mathcal{K} = \left| \frac{x^* f'(x^*)}{f(x^*)} \right|$$

called the *condition number*. If $\mathcal{K}$ is nearly 1, then the error in $f(x)$ is called natural. If it is very large however, then the relative error in the evaluation of $f(x)$ will also be large.

Many numerical methods, especially in linear algebra, involve summations. Quantifying the error propagation in summations becomes an important topic of study. Consider the computation of the sum

$$S = \sum_{i=1}^{m} x_i,$$

where x_i for $i \in 1 : m$ are floating point numbers. Define

$$S_2 = \text{fl}(x_1 + x_2) = (x_1 + x_2)(1 + \varepsilon_2)$$

and recusrively continue

$$S_{r+1} = \text{fl}(S_r + x_{r+1}).$$

Expanding out the nth sum and neglecting $\varepsilon_i\varepsilon_j$ for $i, j \in 1 : n$, we get

$$S_n - (x_1 + x_2 + \cdots + x_n) \approx \sum_{i=1}^{n} \varepsilon_i \sum_{j=1}^{i} x_i = \sum_{j=1}^{n} x_n \sum_{i=j}^{n} \varepsilon_n.$$

Example 1.8. Evaluate the polynomial

$$f(x) = 1.107x^3 + 0.3129x^2 - 0.0172x + 1.1075$$

for $x = 0.1234$ in nested form using 5-digit floating point arithmetic with rounding.

Solution. The polynomial is bracketed as

$$f(x) = ((1.107x + 0.3129)x - 0.0172)x + 1.1075$$

and evaluated from the inside-out according to the rules of 5-digit floating point arithmetic. This yields

$$f(0.1234) \approx 1.1122.$$

■

2 Finding Roots for Nonlinear Equations

Finding the roots of

$$f(x) = 0$$

for an arbitrary f (polynomial or transcendental function) is critical to many scientific and engineering fields. This section talks about various methods to find roots.

Definition 2.1. A function $f : R \to \mathbb{R}$ is said to be a real *polynomial* if

$$f(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x^1 + a_0$$

for a fixed $n \in \mathbb{N}$, called its *degree.*

A real polynomial always has n complex roots, up to repitition.

Definition 2.2. A function $f : X \to \mathbb{R}$ is said to be *transcendental* if its closed form contains trigonometric, exponential, or logarthmic functions.

It is possible to construct algorithms that find roots of transcendental or polynomial equations, especially when it is not possible or impractical to find closed form roots. These algorithms are divided into two categories:

(a) simple enclosure methods, and

(b) fixed point iteration schemes.

Simple closure methods are based on the intermediate value theorem (IVT) described below, and work by finding an interval that surely contains a root and shrinking it systematically.

Theorem 2.3 (Intermediate Value Theorem). *Let f be a continuous function on $[a, b]$, and let $K \in \mathbb{R}$ be a real number in between $f(a)$ and $f(b)$. Then, there exists $c \in (a, b)$ such that*

$$f(c) = K.$$

2.1 Bisection Method

The simplest method is to cut the interval in half, determine which new subinterval contains a root utilizing the IVT, and repeat the process to a desired accuracy. This is fundamentally the *bisection method.*

Theorem 2.4. *Let f be a continuous function on $[a, b]$, and suppose $f(a)f(b) < 0$. The bisection method loosely defined above generates a sequence $\{p_n\} \to p \in (a, b)$ such that $f(p) = 0$ that satisfies*

$$|p_n - p| \leq \frac{b-a}{2^n}.$$

Since $2^{-n} \to 0$, it follows that $\{p_n\} \to p$. We omit the formal proof here due to triviality, and focus on analyzing the *rate of convergence.*

Definition 2.5. Let $\{p_n\} \to p$. If there exists a sequence $\{\beta\} \to 0$ and a positive constant λ such that

$$|p_n - p| \leq \lambda |\beta_n|,$$

for sufficiently large values of n, then $\{p_n\}$ is said to converge to p with *rate of convergence* $O(\beta_n)$.

It is important for all algorithms to possess a stopping criterion, i.e. they need to end in a finite number of steps. The following implementation hard codes this cap if the root isn't within the tolerance range.

```
BISECTION METHOD $(f, a, b, tol, N)$
if $f(a) \cdot f(b) \geq 0$
    error "The root is not bracketed."
for $j = 1$ to $N$
    $c = (a+b)/2$
    if $|f(c)| < tol$
        return $c$ // Convergence criterion met
    if $f(a) \cdot f(c) < 0$
        $b = c$ // Root is in left half
    else $a = c$ // Root is in right half
return $c$ // Best approximation after $N$ iterations
```

Bracketing refers to bounding the root between a positive and negative number.