Round-off/Truncation Errors & Numerical Cancellation

Outline Notes

MS201 — Numerical Analysis I

6 November 2003

1 Introduction

When one tries to classify the errors in numerical computations, it might be useful to study the sources of the errors and the growth of the individual errors. The sources of the errors are essentially static while the growth takes place dynamically. There are essentially three sources of errors:

(a) Initial Errors;

(b) Local round-off errors;

(c) Local truncation errors.

The initial errors are errors in the initial data, a simple example of which is when data are obtained from a physical or chemical apparatus.

Round-off errors depend on the fact that practically each number in a numerical computation must be rounded (or chopped) to a certain number of digits.

Truncation errors arise when an infinite process (in some sense) is replaced by a finite one. Examples of this include the computation of a definite integral through approximation by a sum or the numerical integration of an ordinary differential equation by some finite difference method.

2 Absolute & Relative Errors

Definition 2.1 *Absolute Error: The absolute value of the difference between the number x and its finite representation $fl(x)$.*

$$a.e. = |x - fl(x)|$$

Definition 2.2 *Relative Error: The ratio of the absolute error and the number x:*

$$r.e. = \frac{|x - fl(x)|}{|x|} \geq 0$$

When does one use one or the other definition of error? Consider $x = 0.33$ and $\text{fl}(x) = 0.30$. Clearly, $a.e. = 0.03$ and $r.e. = 0.03/0.33 = .09091 \approx 9.1\%$. When $x = 0.33 \times 10^{-5}$ and $\text{fl}(x) = 0.30 \times 10^{-5}$, $a.e. = 0.03 \times 10^{-5} = 3 \times 10^{-7}$ but $r.e. = 0.09091$. Note that the relative error is unchanged, while the absolute error changed by a factor of 10^5.

Several conclusions can be drawn:

- The absolute error is strongly dependent on the magnitude of x.

- The absolute error is misleading unless it is stated what it is an error of.
- The relative error is a measure of the number of significant digits of x that are correct.
- A relative error has meaning even when x is not known. It is given as a percentage value.

3 Round-off Error

When a calculator or digital computer is used to perform numerical calculations, an unavoidable error, called round-off error, must be considered.

This error arises because the arithmetic performed in a machine involves numbers with only a finite number of digits, with the result that many calculations are performed with approximate representations of the actual numbers. There are two major approaches: chopping and rounding:

- When chopping a number to a specified number of decimal places, say m, the first m digits of the mantissa are retained, simply chopping off the remainder.
- When rounding a number, the computer chooses the closest number that is representable by the computer.

A natural question on may ask is what error is committed when a number is chopped or rounded to n digits? Consider the number

$$x = 0.d_1d_2 \cdots d_nd_{n+1} \cdots$$

Chopping to n digits produces the number

$$\text{fl}(x) = 0.d_1d_2 \cdots d_n$$

with an error

$$\begin{aligned} x - \text{fl}(x) &= 0.d_{n+1}d_{n+2} \cdots \times 10^{-n} \\ &< 0.9999999 \cdots \times 10^{-n} \\ &< 10^{-n} \end{aligned}$$

Does rounding x to n digits increase or decrease this error? We now show that the error is decreased. Once again, consider:

$$x = 0.d_1d_2 \cdots d_nd_{n+1} \cdots$$

If $d_{n+1} < 5$, chop x to n digits. The error is

$$\begin{aligned} x - \text{fl}(x) &= 0.d_{n+1}d_{n_2} \cdots \times 10^{-n} \\ &\leq 0.4999999 \cdots \times 10^{-n} \\ &< 0.5 \times 10^{-n} \end{aligned}$$

When $5 < d_{n+1} \leq 9$, add 0.5×10^{-n} to x and chop the result. If $x^* = x + 0.5 \times 10^{-n}$,

$$\text{round}(x) = \text{chop}(x^*) = 0.d_1d_2 \cdots d_n^*$$

where $0 < d_n^* < 5$ and $d_n^* = d_n + 1$. The error is therefore

$$\begin{aligned} |x - \text{round}(x)| &= 0.d_1d_2\cdots d_n^* - 0.d_1d_2d_3\cdots d_nd_{n+1}\cdots| \\ &= |d_n^* - d_n - 0.d_{n+1}d_{n+2}\cdots| \times 10^{-n} \\ &= |1 - 0.d_{n+1}d_{n+2}\cdots| \times 10^{-n} \\ &< 0.5 \times 10^{-n} \end{aligned}$$

The last inequality follows since $5 < d_{n+1} \leq 9$.

In summary, the error committed by rounding a normalized number x to n digits (in base 10) is 0.5×10^{-n}, which is 1/2 the maximum error committed by chopping.

Example 3.1 *Following Cheney & Kincaid (1985), consider a binary computer having a word length of 64 bits (binary digits) where as much as possible of the normalized floating-point number $\pm q \times 2^m$ must be contained in these 64 bits. One allocation is as follows:*

sign of q	*1 bit*
sign of m	*1 bit*
integer $\|m\|$	*14 bits*
number q	*48 bits*

With such a machine, real numbers with $|m|$ as large as $2^{14} - 1 = 16383$ can be represented. The mantissa is 48 bits and, since $2^{-48} \approx 0.3553 \times 10^{-14}$, it can be inferred that approximately 14 digits of accuracy should be obtained in single precision. For integer representation, only the mantissa is used so that the decimal integers range from $-(2^{48} - 1)$ to $+(2^{48} - 1)$.

This hypothetical machine would represent a machine number as follows. The leftmost bit is used for the sign of the mantissa; the next bit designates the sign of the exponent (0 corresponds to a plus and 1 to a minus sign); the next 14 bits are used to represent the exponent and the remaining 48 bits represent the normalized mantissa.

Consider a positive number in normalized floating-point form,

$$x = \pm q \times 2^m, \qquad \frac{1}{2} \leq q < 1, \qquad |m| \leq 2^{14} - 1 = 16383.$$

Replacing x by its nearest machine number, denoted by $fl(x)$, is a process known as rounding and the error involved is known as round-off error. Writing x in normalized binary notation

$$x = (0.1a_2a_3\cdots a_{48}a_{49}a_{50}\cdots)_2 \times 2^m$$

the nearest machine number to the left of x (rounding down) on the real number line is

$$x_L = (0.1a_2a_3\cdots a_{48})_2 \times 2^m$$

while the nearest machine number to the right of x (rounding up) is

$$x_R = \left[(0.1a_2a_3\cdots a_{48})_2 + 2^{-48}\right] \times 2^m.$$

The closer of these machine numbers is the one chosen to represent x.

If x lies closer to x_L than x_R, then

$$|x - x_L| \leq \frac{1}{2}|x_R - x_L| = 2^{-49+m},$$

so that the relative error bound is calculated as

$$\frac{|x - x_L|}{|x|} \leq \frac{2^{-49+m}}{(0.1a_2a_3\cdots)_2 \times 2^m} \leq \frac{2^{-49}}{\frac{1}{2}} = 2^{-48},$$

and the same result applies when x is closer to x_R. The number 2^{-48} is known as the unit round-off error for this hypothetical machine.

More generally, let a real number x be written in the form

$$x = s \cdot (.d_1 d_2 \cdots d_n d_{n+1} \cdots)_\beta \times \beta^m$$

The chopped machine representation of x is given by

$$\text{fl}(x) = s \cdot (.d_1 d_2 \cdots d_n)_\beta \times \beta^m$$

The rounded machine representation of x is given by

$$\text{fl}(x) = \begin{cases} s \cdot (.d_1 d_2 \cdots d_n)_\beta \times \beta^m & 0 \le a_{n+1} < \frac{\beta}{2} \\ s \cdot \left[(.d_1 d_2 \cdots d_n)_\beta + (.00 \cdots 01)_\beta \right] \times \beta^m & \frac{\beta}{2} \le a_{n+1} < \beta \end{cases}$$

Note that $(.00 \cdots 01)_\beta$ denotes β^{-n}. With most real numbers, we have $\text{fl}(x) \neq x$ and it is easy to show that the relative error is

$$\frac{x - \text{fl}(x)}{x} = -\epsilon,$$

where

$$\begin{array}{ll} -\beta^{-n+1} \le \epsilon \le 0 & \text{chopped fl}(x) \\ -\frac{1}{2}\beta^{-n+1} \le \epsilon \le \frac{1}{2}\beta^{-n+1} & \text{rounded fl}(x) \end{array}$$

Following Atkinson (1989), we will illustrate the proof for chopping. Assume that $s = +1$ since the case $s = -1$ will not alter the sign of ϵ. Since

$$x = (.d_1 d_2 \cdots d_n d_{n+1} \cdots)_\beta \times \beta^m$$

and the chopped machine representation of x is

$$\text{fl}(x) = (.d_1 d_2 \cdots d_n)_\beta \times \beta^m$$

we have

$$x - \text{fl}(x) = (.00 \cdots 0 d_{n+1} \cdots)_\beta \times \beta^m$$

Letting $\gamma = \beta - 1$, we obtain

$$\begin{aligned} 0 &\le x - \text{fl}(x) \\ &\le (.00 \cdots 0 \gamma\gamma \cdots)_\beta \times \beta^m \\ &= \gamma \left[\beta^{-n-1} + \beta^{-n-2} + \cdots \right] \times \beta^m \\ &= \gamma \left[\frac{\beta^{-n-1}}{1 - \beta^{-1}} \right] \times \beta^m \\ &= \beta^{-n+m} \end{aligned}$$

Therefore, the relative error for chopping is obtained from

$$\begin{aligned} 0 &\le \frac{x - \text{fl}(x)}{x} \\ &\le \frac{\beta^{-n+m}}{(.d_1 d_2 \cdots d_n d_{n+1} \cdots)_\beta} \\ &\le \frac{\beta^{-n}}{(.100 \cdots)_\beta} \\ &\le \beta^{-n+1} \end{aligned}$$

The formula for the relative error

$$\frac{x - \mathrm{fl}(x)}{x} = -\epsilon,$$

is usually written as

$$\mathrm{fl}(x) = (1+\epsilon)x.$$

Definition 3.1 *The "unit round" of a computer is the number δ satisfying: (a) it is a positive floating-point number, and (b) it is the smallest such number for which*

$$fl(1+\delta) = 1.$$

The unit round δ is given by

$$\delta = \begin{array}{ll} \beta^{-n+1} & \text{chopped definition of } \mathrm{fl}(x) \\ \frac{1}{2}\beta^{-n+1} & \text{rounded definition of } \mathrm{fl}(x) \end{array}$$

Example 3.2 *Consider rounding arithmetic on a binary machine. In order to show that*

$$fl(1+2^{-n}) > 1,$$

write

$$\begin{aligned} 1 + 2^{-n} &= [(.100\cdots)_2 + (.000\cdots00100\cdots)_2] \times 2^1 \\ &= (.100\cdots00100\cdots)_2 \times 2^1. \end{aligned}$$

Form $fl(1+2^{-n})$, noting that there is a 1 in the $(n+1)$-st position of the mantissa,

$$fl(1+2^{-n}) = (.100\cdots01)_2 \times 2^1 \;=\; 1 + 2^{-n+1},$$

and so

$$fl(1+2^{-n}) > 1,$$

although

$$fl(1+2^{-n}) \neq 1 + 2^{-n}.$$

The fact that δ cannot be smaller than 2^{-n} follows by considering again

$$1 + 2^{-n} = (.100\cdots00100\cdots)_2 \times 2^1.$$

If $\hat{\delta} < \delta$, then $1 + \hat{\delta}$ has a zero in the $(n+1)$-st position of the mantissa and the definition of rounding would then imply that $fl(1+\hat{\delta}) = 1$.

Definition 3.2 *Exact Integers Bound: is the largest integer K for which*

$$k \text{ an integer and } 0 \leq k \leq K \Rightarrow fl(k) = k.$$

This also implies that $fl(K+1) \neq K+1$. Show that $K = \beta^m$.

Noting that the lower (L) and upper (U) exponent limits satisfy $L \leq m \leq U$, the smallest positive floating-point number is

$$x_L = (.100\cdots0)_\beta \times \beta^L.$$

Using the notation $\gamma = \beta - 1$, the largest positive floating-point number is

$$\begin{aligned} x_U &= (.\gamma\gamma\cdots\gamma)_\beta \times \beta^U \\ &= \left(1 - \beta^{-m}\right) \times \beta^U. \end{aligned}$$

Thus, all floating-point numbers x must satisfy

$$x_L \;\leq\; x \;\leq\; x_U.$$

4 Significant digits

Our intuition tells us that the more accurate a number is, the more digits (that follow the decimal point) will be correct, once the number is expressed in normalized format. To make this statement more precise, we relate the number of correct digits to the relative error.

Definition 4.1 *p^* approximates p to t significant digits if t is the largest integer > 0 such that*

$$r.e. = \frac{|p - p^*|}{p} \leq 5 \times 10^{-t}$$

Taking $\log_{10}$ of both sides results in

$$\begin{aligned} \log_{10}(r.e.) &\leq \log_{10}(5 \times 10^{-t}) \\ &= \log_{10} 5 - t \end{aligned}$$

so that

$$t \leq \log_{10} \frac{5}{r.e.}$$

A related statement is that if x and y are normalized floating-point machine numbers such that $x > y > 0$ and $1 - \frac{y}{x} \geq 2^{-k}$, then at most k significant binary bits are lost in the subtraction $x - y$.

Example 4.1 *Given a relative error $r.e. = 0.5$, how many significant digits do we have?*

$$t \leq \log_{10} \frac{5}{5 \times 10^{-1}} = 1$$

Given $r.e. = 0.1$,

$$t \leq \log_{10} \frac{5}{0.1} = \log_{10} 50$$

so that $1 < t < 2$. There is one significant digit.

Example 4.2 *Consider $x = 3.29$ and $fl(x) = 3.2$. How accurate is the approximation $fl(x)$? The relative error is*

$$r.e. = \left| \frac{3.29 - 3.2}{3.29} \right| = \frac{9 \times 10^{-2}}{3.29} \approx 3 \times 10^{-2}$$

so that

$$t \leq \log_{10} \frac{5}{3 \times 10^{-2}} = 2 + \log_{10}(5/3)$$

which implies that t lies between 2 and 3. There are 2 significant digits.

The number of significant digits is only weakly dependent on the value of the relative error mantissa. For example, as long as

$$r.e. \in [0.5 \times 10^{-2},\ 5.0 \times 10^{-2}]$$

there are two significant digits.

Example 4.3 *Another way to pose the question is given $x = 3.2$, what is the worst possible approximation to x which is accurate to 2 significant digits? Let the approximation be x^*. Therefore*

$$2 = t = \log_{10} \frac{5}{\left| \frac{3.2 - x^*}{3.2} \right|}$$

Taking the antilog of both sides (10 to the power),

$$100 = \frac{5}{\left|\frac{3.2-x^*}{3.2}\right|}$$

Rearranging terms,

$$\frac{20}{3.2} = |3.2 - x^*|^{-1}$$

or

$$|3.2 - x^*| \approx \frac{1}{6.25} = 0.16$$

The worst approximation to x while retaining 2 significant digits is therefore $x^* = 3.04$ or $x^* = 3.36$.

The last example demonstrates that the concept of significant digits is not simply a matter of counting the number of digits which are correct.

5 Floating Point Operations

Example 5.1 *Let $x = 3.3453236$ and $y = 2.316236236$. The addition of x and y is*

$$x + y = 5.661559836$$

which is assumed to be exact (it is in this example). Now assume 4 digit arithmetic (normalized form).

$$fl(x) = 0.3345 \times 10^1$$

and

$$fl(y) = 0.2316 \times 10^1$$

so that the addition is

$$fl(x) + fl(y) = 0.5661 \times 10^1.$$

As a final step, the computer rounds this number to 4 digits by applying the $fl()$ operator to both sides:

$$x \oplus y = fl\{fl(x) + fl(y)\} = 0.5661 \times 10^1$$

The intermediate operation $fl(x) + fl(y)$ is assumed for the sake of exposition to be performed exactly. In reality, the computer must perform this operation using one or more words of finite length.

Example 5.2 *Consider 5 digit chopping (all numbers are normalized).*

$$x = 1/3, \qquad y = 5/7$$

so that

$$fl(x) = 0.33333 \times 10^0 \qquad \text{and} \qquad fl(y) = 0.71428 \times 10^0.$$

Let us add two positive numbers.

$$x \oplus y = 0.10476 \times 10^1 = fl\{fl(x) + fl(y)\}$$

However, the exact value is

$$\begin{aligned} x+y &= \frac{1}{3}+\frac{5}{7} \\ &= \frac{22}{21} \\ &= 1.047610\cdots \\ &= 0.1047619\times 10^{1} \end{aligned}$$

The absolute error is

$$\begin{aligned} a.e. &= (0.1047619-0.10476)\times 10^{1} \\ &= 0.0000019\times 10^{1} \\ &= 0.19\times 10^{-4} \end{aligned}$$

The relative error is

$$r.e. = \frac{0.19\times 10^{-4}}{1.04\cdots} \approx 0.2\times 10^{-4}$$

while the number of significant digits t satisfies

$$\begin{aligned} t &< \log_{10}\frac{5}{.2\times 10^{-4}} \\ &= \log_{10} 25+\log_{10} 10^{4} \\ &< \log_{10} 2.5+5 \end{aligned}$$

Therefore,

$$t=5$$

6 Error Propagation

Assume the operation

$$x\oplus y=(1+\delta)(x+y)$$

δ is the relative error of this operation. Errors propagate across multiple operations.

For example, the addition of three numbers:

$$\begin{aligned} (x\oplus y)\oplus z &= \mathrm{fl}\,\{x(1+\delta)+y(1+\delta))+z(1+\delta)\} \\ &= (1+3\delta)(x+y+z) \end{aligned}$$

We have assumed that all the relative errors are equal. The result is that the addition of three numbers gives a maximum relative error of three times the r.e. of one of the input numbers. In reality, different signs for the error lead to error cancellations, so that the results are not so detrimental.

Example 6.1 *Consider the evaluation of the integral*

$$I_n=\frac{1}{e}\int_0^1 e^x x^n\,dx$$

A little manipulation (integration by parts) yields the recurrence relation

$$\begin{aligned} I_0 &= 1 - \frac{1}{e} \\ I_n &= 1 - nI_{n-1}, \qquad n = 1, 2, \ldots \end{aligned}$$

Note that the integrand is always positive within the range of integration and that the area under the curve, and hence the value of I_n, decreases monotonically with n. Thus, for all n, we may deduce that $I_n > 0$ and $I_n < I_{n-1}$.

The results of running a Matlab program implementing this recurrence relation in double-precision (16-digit) arithmetic are as shown in the following table:

n	*Exact Value*	*Recursion*	*Relative Error*
0	$6.321206e-01$	$6.321206e-01$	$0.00e+00$
1	$3.678794e-01$	$3.678794e-01$	$0.00e+00$
2	$2.642411e-01$	$2.642411e-01$	$0.00e+00$
3	$2.072766e-01$	$2.072766e-01$	$2.68e-16$
4	$1.708934e-01$	$1.708934e-01$	$1.79e-15$
5	$1.455329e-01$	$1.455329e-01$	$1.03e-14$
6	$1.268024e-01$	$1.268024e-01$	$7.11e-14$
7	$1.123835e-01$	$1.123835e-01$	$5.57e-13$
8	$1.009320e-01$	$1.009320e-01$	$4.97e-12$
9	$9.161229e-02$	$9.161229e-02$	$4.92e-11$
10	$8.387707e-02$	$8.387707e-02$	$5.38e-10$
11	$7.735223e-02$	$7.735223e-02$	$6.41e-09$
12	$7.177325e-02$	$7.177325e-02$	$8.29e-08$
13	$6.694770e-02$	$6.694778e-02$	$1.16e-06$
14	$6.273216e-02$	$6.273108e-02$	$1.73e-05$
15	$5.901754e-02$	$5.903379e-02$	$2.75e-04$
16	$5.571935e-02$	$5.545930e-02$	$4.67e-03$
17	$5.277112e-02$	$5.719187e-02$	$8.38e-02$
18	$5.011985e-02$	$-2.945367e-02$	$1.59e+00$
19	$4.772276e-02$	$1.559620e+00$	$3.17e+01$
20	$4.554488e-02$	$-3.019239e+01$	$6.64e+02$
21	$4.355743e-02$	$6.350403e+02$	$1.46e+04$
22	$4.173644e-02$	$-1.396989e+04$	$3.35e+05$
23	$4.006181e-02$	$3.213084e+05$	$8.02e+06$
24	$3.851655e-02$	$-7.711400e+06$	$2.00e+08$
25	$3.708621e-02$	$1.927850e+08$	$5.20e+09$
26	$3.575842e-02$	$-5.012410e+09$	$1.40e+11$
27	$3.452253e-02$	$1.353351e+11$	$3.92e+12$
28	$3.336929e-02$	$-3.789382e+12$	$1.14e+14$
29	$3.229068e-02$	$1.098921e+14$	$3.40e+15$
30	$3.127967e-02$	$-3.296762e+15$	$1.05e+17$

For large n, the values do not exhibit the anticipated behaviour. For $n > N_p$, where N_p is dependent on the precision of the arithmetic, successive values of I_n increase in magnitude and alternate in sign.

Increasing the precision merely delays the point at which the problems are noticed. Why does this happen? The computation of I_0 is not exact; no matter how good the intrinsic functions are, there will be an error, ϵ_0, in the value we obtain for I_0. We therefore start our recurrence relation with $\hat{I}_0$ where

$$\hat{I}_0 = I_0 + \epsilon_0.$$

Even if we incur no further rounding errors, the effect of this initial error is such that we compute a sequence

$$\left\{\hat{I}_n \mid n = 1, 2, \ldots\right\}$$

using the recurrence relation

$$\hat{I}_n = 1 - n\hat{I}_{n-1}, \qquad n = 1, 2, \ldots,$$

rather than the values $\{I_n\}$ defined by

$$I_n = 1 - nI_{n-1}, \qquad n = 1, 2, \ldots.$$

If we let $\epsilon_n = \hat{I}_n - I_n$ be the error at step n, then, using the two recurrence relations, we obtain

$$\epsilon_n = -n\epsilon_{n-1},$$

and hence

$$\epsilon_n = (-1)^n\, n!\, \epsilon_0.$$

Using double-precision arithmetic, $\epsilon_0 \approx 2.204 \times 10^{-16}$ and the rapid growth of the factorial is enough to destroy any accuracy we might hope to achieve. We say that this recurrence relation is **unstable**, *and it is important to recognize such situations, as well as those which are inherently stable.*

7 Numerical Cancellation

Example 7.1 *Consider the quadratic equation $x^2 - 2ax + \epsilon = 0$ which has the two solutions*

$$x_1 = a + \sqrt{a^2 - \epsilon} \qquad \text{and} \qquad x_2 = a - \sqrt{a^2 - \epsilon}.$$

If $a > 0$ and ϵ is small compared with a, the root x_2 is expressed as the difference between two almost equal numbers and a considerable amount of significance is lost. Instead, if we write

$$x_2 = \frac{\epsilon}{a + \sqrt{a^2 - \epsilon}},$$

we obtain the root as approximately $\frac{\epsilon}{2a}$ without loss of significance.

Example 7.2 *Suppose that, for a fairly large value of x, we know that $\cosh(x) = a$; $\sinh(x) = b$ and that we want to compute e^{-x}. Clearly*

$$e^{-x} = \cosh(x) - \sinh(x) = a - b,$$

leading to a dangerous cancellation while, on the other hand,

$$e^{-x} = \frac{1}{\cosh(x) + \sinh(x)} = \frac{1}{a + b},$$

gives a very accurate result.

8 Some Final Thoughts

Suppose we want to compute $f(x)$ where x is a real number and f is a real function. In practical computations, the number x must be approximated by a rational number r since no computer can store numbers with an infinite number of decimals.

The difference $|r - x|$ constitutes the *initial error* while the difference $\epsilon_0 = |f(r) - f(x)|$ is the corresponding *propagated error*. In many cases, f is such a function that it must be replaced by a simpler function f_1 (often a truncated power series expansion of f).

The difference $\epsilon_1 = |f_1(r) - f(r)|$ is then the *truncation error*. The calculations performed by the computer, however, are not exact but pseudo-operations of a type discussed earlier. The result is that instead of getting $f_1(r)$ we get another value $f_2(r)$ which is then a wrongly computed value of a wrong function of a wrong argument.

The difference $\epsilon_2 = |f_2(r) - f_1(r)|$ could be termed the propagated error from the roundings. The total error is $\epsilon = \epsilon_0 + \epsilon_1 + \epsilon_2$.