Chapter One: Mathematical Preliminaries and Error Analysis

Numerical Analysis is a field of mathematics that concerned with the study of approximate solutions of mathematical problems, where it is difficult or impossible to find the exact solutions for these problems.

For instance, we can't find the exact value of the following integral, by using known integration methods,

$$\int_0^1 e^{x^2}\, dx$$

While we can find an approximate value for this integral, using numerical methods.

The other example, if we aim to find the solution of a linear system $Ax = b,$ where $A \in R^{n\times n}$, when n is too large , it is so difficult to find the exact solutions for this system by hand using known methods, so in this case, it is easier to think about how to find the approximate solution using a suitable algorithm and computer programs.

The importance of Numerical Analysis:

To interpret any real phenomena, we need to formulate it in a mathematical form. To give a realistic meaning for these phenomena we have to choose complicated mathematical models, but the problem is; it is so difficult to find explicit formulas describe the exact solutions for these complicated models. Therefore, it might be better to find the numerical solutions for complicated form rather than finding the exact solutions for easier forms that can't describe these phenomena in realistic way.

The nature of Numerical analysis:

Since for any numerical Algorithm (the steps of the numerical method), we have lots of mathematical calculations, we need to choose a suitable computer language

Round-off Errors and Computer Arithmetic:

The arithmetic performed by a calculator or computer is different from the arithmetic in algebra and calculus courses. You would likely expect that we always have as true statements things such as $2+2=4$, $4\cdot 8=32$, and $(\sqrt{3})^2=3$. However, with *computer* arithmetic we expect exact results for $2+2=4$ and $4\cdot 8=32$, but we will not have precisely $(\sqrt{3})^2=3$. To understand why this is true we must explore the world of finite-digit arithmetic.

In our traditional mathematical world we permit numbers with an infinite number of digits. The arithmetic we use in this world *defines* $\sqrt{3}$ as that unique positive number that when multiplied by itself produces the integer 3. In the computational world, however, each representable number has only a fixed and finite number of digits. This means, for example, that only rational numbers—and not even all of these—can be represented exactly. Since $\sqrt{3}$ is not rational, it is given an approximate representation, one whose square will not be precisely 3, although it will likely be sufficiently close to 3 to be acceptable in most situations. In most cases, then, this machine arithmetic is satisfactory and passes without notice or concern, but at times problems arise because of this discrepancy.

The error that is produced when a calculator or computer is used to perform real-number calculations is called **round-off error**. It occurs because the arithmetic performed in a machine involves numbers with only a finite number of digits, with the result that calculations are performed with only approximate representations of the actual numbers. In a computer, only a relatively small subset of the real number system is used for the representation of all the real numbers. This subset contains only rational numbers, both positive and negative, and stores the fractional part, together with an exponential part.

Binary Machine Numbers:

In 1985, the IEEE (Institute for Electrical and Electronic Engineers) published a report called *Binary Floating Point Arithmetic Standard 754–1985*. An updated version was published in 2008 as *IEEE 754-2008*. This provides standards for binary and decimal floating point numbers, formats for data interchange, algorithms for rounding arithmetic operations, and for the handling of exceptions. Formats are specified for single, double, and extended precisions, and these standards are generally followed by all microcomputer manufacturers using floating-point hardware.

Decimal Machine Numbers:

The use of binary digits tends to conceal the computational difficulties that occur when a finite collection of machine numbers is used to represent all the real numbers. To examine these problems, we will use more familiar decimal numbers instead of binary representation. Specifically, we assume that machine numbers are represented in the normalized *decimal* floating-point form

$$\pm 0.d_1d_2\ldots d_k \times 10^n, \quad 1 \le d_1 \le 9, \quad \text{and} \quad 0 \le d_i \le 9,$$

for each $i = 2, \ldots, k$. Numbers of this form are called k-digit *decimal machine numbers*.

Any positive real number within the numerical range of the machine can be normalized to the form

$$y = 0.d_1d_2\ldots d_kd_{k+1}d_{k+2}\ldots \times 10^n.$$

The floating-point form of y, denoted $fl(y)$, is obtained by terminating the mantissa of y at k decimal digits. There are two common ways of performing this termination. One method, called **chopping**, is to simply chop off the digits $d_{k+1}d_{k+2}\ldots$. This produces the floating-point form

$$fl(y) = 0.d_1d_2\ldots d_k \times 10^n.$$

The other method, called **rounding**, adds $5 \times 10^{n-(k+1)}$ to y and then chops the result to obtain a number of the form

$$fl(y) = 0.\delta_1\delta_2\ldots\delta_k \times 10^n.$$

For rounding, when $d_{k+1} \ge 5$, we add 1 to d_k to obtain $fl(y)$; that is, we *round up*. When $d_{k+1} < 5$, we simply chop off all but the first k digits; so we *round down*. If we round down, then $\delta_i = d_i$, for each $i = 1, 2, \ldots, k$. However, if we round up, the digits (and even the exponent) might change.

Note: The error that result from replacing a number with its floating-point form is called ***round-off error*** regardless of whether the rounding or chopping method is used.

Example:

Determine the five-digit **(a)** chopping and **(b)** rounding values of the irrational number π.

Solution The number π has an infinite decimal expansion of the form $\pi = 3.14159265\ldots$. Written in normalized decimal form, we have

$$\pi = 0.314159265\ldots \times 10^1.$$

(a) The floating-point form of π using five-digit chopping is

$$fl(\pi) = 0.31415 \times 10^1 = 3.1415.$$

(b) The sixth digit of the decimal expansion of π is a 9, so the floating-point form of π using five-digit rounding is

$$fl(\pi) = (0.31415 + 0.00001) \times 10^1 = 3.1416. \quad ■$$

The following definition describes two methods for measuring approximation errors.

Definition:

Suppose that p^* is an approximation to p. The **absolute error** is $|p - p^*|$, and the **relative error** is $\dfrac{|p - p^*|}{|p|}$, provided that $p \neq 0$. ■

Example:

Determine the absolute and relative errors when approximating p by p^* when

(a) $p = 0.3000 \times 10^1$ and $p^* = 0.3100 \times 10^1$;

(b) $p = 0.3000 \times 10^{-3}$ and $p^* = 0.3100 \times 10^{-3}$;

(c) $p = 0.3000 \times 10^4$ and $p^* = 0.3100 \times 10^4$.

Solution

(a) For $p = 0.3000 \times 10^1$ and $p^* = 0.3100 \times 10^1$ the absolute error is 0.1, and the relative error is $0.333\overline{3} \times 10^{-1}$.

(b) For $p = 0.3000 \times 10^{-3}$ and $p^* = 0.3100 \times 10^{-3}$ the absolute error is 0.1×10^{-4}, and the relative error is $0.333\overline{3} \times 10^{-1}$.

(c) For $p = 0.3000 \times 10^4$ and $p^* = 0.3100 \times 10^4$, the absolute error is 0.1×10^3, and the relative error is again $0.333\overline{3} \times 10^{-1}$.

Finite-Digit Arithmetic:

Assume that the floating-point representations $fl(x)$ and $fl(y)$ are given for the real numbers x and y and that the symbols $\oplus, \ominus, \otimes, \oplus$ represent machine addition, subtraction, multiplication, and division operations, respectively. We will assume a finite-digit arithmetic given by

$$x \oplus y = fl(fl(x) + fl(y)), \quad x \otimes y = fl(fl(x) \times fl(y)),$$
$$x \ominus y = fl(fl(x) - fl(y)), \quad x \oplus y = fl(fl(x) \div fl(y)).$$

This arithmetic corresponds to performing exact arithmetic on the floating-point representations of x and y and then converting the exact result to its finite-digit floating-point representation.

Example:

Suppose that $x = \frac{5}{7}$ and $y = \frac{1}{3}$. Use five-digit chopping for calculating $x + y$, $x - y$, $x \times y$, and $x \div y$.

Solution Note that x= 0.714285714285714285.... y=0.3333333333....

$$x = \frac{5}{7} = 0.\overline{714285} \quad \text{and} \quad y = \frac{1}{3} = 0.\overline{3}$$

implies that the five-digit chopping values of x and y are

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

Thus

$$x \oplus y = fl(fl(x) + fl(y)) = fl\left(0.71428 \times 10^0 + 0.33333 \times 10^0\right)$$
$$= fl\left(1.04761 \times 10^0\right) = 0.10476 \times 10^1.$$

The true value is $x + y = \frac{5}{7} + \frac{1}{3} = \frac{22}{21}$, so we have

$$\text{Absolute Error} = \left|\frac{22}{21} - 0.10476 \times 10^1\right| = 0.190 \times 10^{-4}$$

and

$$\text{Relative Error} = \left|\frac{0.190 \times 10^{-4}}{22/21}\right| = 0.182 \times 10^{-4}.$$

Operation	Result	Actual value	Absolute error	Relative error
$x \oplus y$	0.10476×10^1	$22/21$	0.190×10^{-4}	0.182×10^{-4}
$x \ominus y$	0.38095×10^0	$8/21$	0.238×10^{-5}	0.625×10^{-5}
$x \otimes y$	0.23809×10^0	$5/21$	0.524×10^{-5}	0.220×10^{-4}
$x \oplus\!\!\!\!\!\div y$	0.21428×10^1	$15/7$	0.571×10^{-4}	0.267×10^{-4}

Example:

Suppose that in addition to $x = \frac{5}{7}$ and $y = \frac{1}{3}$ we have

$$u = 0.714251, \quad v = 98765.9, \quad \text{and} \quad w = 0.111111 \times 10^{-4},$$

so that

$$fl(u) = 0.71425 \times 10^0, \quad fl(v) = 0.98765 \times 10^5, \quad \text{and} \quad fl(w) = 0.11111 \times 10^{-4}.$$

Determine the five-digit chopping values of $x \ominus u$, $(x \ominus u) \oplus\!\!\!\!\!\div w$, $(x \ominus u) \otimes v$, and $u \oplus v$.

Solution These numbers were chosen to illustrate some problems that can arise with finite-digit arithmetic. Because x and u are nearly the same, their difference is small. The absolute error for $x \ominus u$ is

$$\begin{aligned}|(x-u)-(x \ominus u)| &= |(x-u)-(fl(fl(x)-fl(u)))| \\ &= \left|\left(\frac{5}{7}-0.714251\right)-\left(fl\left(0.71428 \times 10^0 - 0.71425 \times 10^0\right)\right)\right| \\ &= \left|0.347143 \times 10^{-4} - fl\left(0.00003 \times 10^0\right)\right| = 0.47143 \times 10^{-5}.\end{aligned}$$

This approximation has a small absolute error, but a large relative error

$$\left|\frac{0.47143 \times 10^{-5}}{0.347143 \times 10^{-4}}\right| \le 0.136.$$

The subsequent division by the small number w or multiplication by the large number v magnifies the absolute error without modifying the relative error. The addition of the large and small numbers u and v produces large absolute error but not large relative error. These calculations are shown in Table **Below:** ■

Operation	Result	Actual value	Absolute error	Relative error
$x \ominus u$	0.30000×10^{-4}	0.34714×10^{-4}	0.471×10^{-5}	0.136
$(x \ominus u) \oplus w$	0.27000×10^{1}	0.31242×10^{1}	0.424	0.136
$(x \ominus u) \otimes v$	0.29629×10^{1}	0.34285×10^{1}	0.465	0.136
$u \oplus v$	0.98765×10^{5}	0.98766×10^{5}	0.161×10^{1}	0.163×10^{-4}

Example:

Let $p = 0.54617$ and $q = 0.54601$. Use four-digit arithmetic to approximate $p - q$ and determine the absolute and relative errors using **(a)** rounding and **(b)** chopping.

Solution The exact value of $r = p - q$ is $r = 0.00016$.

(a) Suppose the subtraction is performed using four-digit rounding arithmetic. Rounding p and q to four digits gives $p^* = 0.5462$ and $q^* = 0.5460$, respectively, and $r^* = p^* - q^* = 0.0002$ is the four-digit approximation to r. Since

$$\frac{|r - r^*|}{|r|} = \frac{|0.00016 - 0.0002|}{|0.00016|} = 0.25,$$

the result has only one significant digit, whereas p^* and q^* were accurate to four and five significant digits, respectively.

(b) If chopping is used to obtain the four digits, the four-digit approximations to p, q, and r are $p^* = 0.5461$, $q^* = 0.5460$, and $r^* = p^* - q^* = 0.0001$. This gives

$$\frac{|r - r^*|}{|r|} = \frac{|0.00016 - 0.0001|}{|0.00016|} = 0.375,$$

which also results in only one significant digit of accuracy. ■

Exercises:

Q1:

Compute the absolute error and relative error in approximations of p by p^*.

a. $p = \pi, p^* = 22/7$

b. $p = \pi, p^* = 3.1416$

c. $p = e, p^* = 2.718$

d. $p = \sqrt{2}, p^* = 1.414$

e. $p = e^{10}, p^* = 22000$

f. $p = 10^\pi, p^* = 1400$

g. $p = 8!, p^* = 39900$

h. $p = 9!, p^* = \sqrt{18\pi}(9/e)^9$

Q2:

Use three-digit rounding arithmetic to perform the following calculations. Compute the absolute error and relative error with the exact value determined to at least five digits.

a. $133 + 0.921$

b. $133 - 0.499$

c. $(121 - 0.327) - 119$

d. $(121 - 119) - 0.327$

e. $\dfrac{\frac{13}{14} - \frac{6}{7}}{2e - 5.4}$

f. $-10\pi + 6e - \dfrac{3}{62}$

g. $\left(\dfrac{2}{9}\right) \cdot \left(\dfrac{9}{7}\right)$

h. $\dfrac{\pi - \frac{22}{7}}{\frac{1}{17}}$

Q3:

Repeat (Q2) using three-digit chopping arithmetic

Q4:

Let E_x, R_x be the absolute and relative errors, respectively, in an approximate value of x. Show that: $R_x \leq E_x$, if $|x| \geq 1$.

Q5:

a. Use three-digit chopping arithmetic to compute the sum $\sum_{i=1}^{10}(1/i^2)$ first by $\frac{1}{1} + \frac{1}{4} + \cdots + \frac{1}{100}$ and then by $\frac{1}{100} + \frac{1}{81} + \cdots + \frac{1}{1}$. Which method is more accurate, and why?

b. Write an algorithm to sum the finite series $\sum_{i=1}^{N} x_i$ in reverse order.