Numerical Methods

Xuemei Chen

New Mexico State University

(Last updated on January 28, 2020)

Contents

Notations

Before this course, you need to be familiar with

- Calculus I
- Calculus II: Integrations. Sequences and Series.
- It is not necessary, but helpful to know some multivariate Calculus
- Some basic Linear Algebra.

Notations

- $\{x|$ descriptions of $x\}$ is the set of all x that meets the description after the ":" sign. For example, $\{x|x^2-1<0\}$ is the set of all number x such that $x^2-1<0$. We can solve this inequality further and see that $\{x|x^2-1<0\}=(-1,1)$

  Remark: Some books use colon instead of verticle bar as $\{x:$ descriptions of $x\}$
- $\in$: belongs to/is contained in. For example, $a\in\{x|x^2-1<0\}$ means that a is a number satisfying $a^2-1<0$.
- A vector in $\mathbb{R}^n$ is a column vector in default.
- In all exercises at the end of each chapter, * means extra credit problems.

Chapter 1

Floating Point Arithmetic

The arithmetic performed by a calculator or computer is different from the arithmetic that we use in our algebra or calculus class. In the ideal world, we permit numbers with an infinite number of digits. For example, $\sqrt{2}$ is the unique positive number that, when multiplied by itself, produces the integer 2. It is an irrational number, which means that there are infinitely many digits associated with $\sqrt{2}$ when represented in decimals. In the computational world, we are only able to assign a fixed and finite number of digits to each number.

Try add up 0.1 and 0.2 in Python, you will get

```
>>> 0.1 + 0.2
0.30000000000000004
```

This can cause serious issues and create bugs. See Section 5.1 of [2]. The above error is caused by rounding in floating point numbers, with which numerical analysis is traditionally concerned.

1.1 Floating Point Representation

Floating point numbers are represented in terms of a base β, a precision p, and an exponent e. For example, in base 10, 0.34 can be represented as 3.4×10^{-1} or 34×10^{-2}. In order to guarantee uniqueness, we take a floating point number to have the specific form.

$$\pm \left(d_0 + d_1\beta^{-1} + \cdots + d_{p-1}\beta^{-(p-1)}\right) \times \beta^e, \tag{1.1}$$

where each d_i is an integer in $[0, \beta)$ and $d_0 \neq 0$. Such a representation is said to be a *normalized floating point number*. In this representation, $d_0 + d_1\beta^{-1} + \cdots + d_{p-1}\beta^{-(p-1)}$ is called *mantissa* (or *significand*). Here are a few examples where $\beta = 10, p = 5$ and e is in the range $-10 \leq e \leq 10$:

$$1.2345 \times 10^8, \qquad 3.4000 \times 10^{-2}, \qquad -5.6780 \times 10^5$$

In the above examples, we are using base $\beta = 10$ to ease into the material. Computers work more naturally in binary (base 2). When $\beta = 2$, each digit d_i is 0 or 1 in equation (1.1). I will leave it to the readers to review binary representation, but some examples are listed below:

- Conversion from binary to decimal:
  a. $11.0101_2 = 1 \cdot 2^1 + 1 \cdot 2^0 + 0 \cdot 2^{-1} + 1 \cdot 2^{-2} + 0 \cdot 2^{-3} + 1 \cdot 2^{-4} = 2 + 1 + 0.25 + 0.0625 = 3.3125$
  b. $1100_2 = 1.1_2 \times 2^3 = (1 \cdot 2^0 + 1 \cdot 2^{-1}) \cdot 2^3 = 12$
  c. $0.01_2 = 1.00_2 \times 2^{-2} = 0.25$

- Conversion from decimal to binary:

  a. $41 = 2^5 + 9 = 1 * 2^5 + 1 * 2^3 + 1 * 2^0 = 101001_2$

  b. To figure out the binary representation for 0.1, we first observe that $2^{-4} = 0.0625$ is the biggest component of 0.1. The leftover $0.1 - 0.0625 = 0.0375 > 2^{-5}$. So $0.1 = 2^{-4} + 2^{-5} + 0.00625 = 2^{-4} + 2^{-5} + 2^{-4} * 0.1 = 2^{-4} + 2^{-5} + 2^{-4}(2^{-4} + 2^{-5} + 2^{-4} * 0.1)$. This can keep going, so

$$0.1 = 2^{-4} + 2^{-5} + 2^{-8} + 2^{-9} + \cdots = 0.0\overline{0011}_2 = 1.\overline{1001}_2 \times 2^{-4}. \tag{1.2}$$

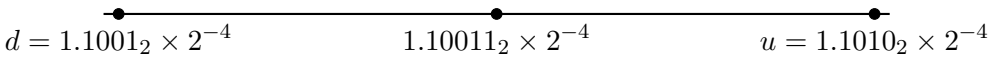
As seen, if a base (subscript) is not specified, then it is understood that it is a decimal number.

The representation in (1.2) has to be rounded at some point, and this is causing the computation inaccuracy at the beginning of this chapter.

There are usually 4 rounding models: rounding down, rounding up, rounding towards 0, and rounding to nearest. rounding to the nearest is, either round down or round up, whichever is closer. In case of a tie, we choose the one whose least significant (rightmost) bit is smaller. Given a real number x, we denote its floating point representation by $\text{fl}(x)$. For convenience of illustration, we use examples where we retain 4 digits after the binary point (this means the representation has precision $p = 5$).

Example 1.1. Round $0.1 = 1.1001100\cdots_2 \times 2^{-4}$ to the nearest so that there are only 4 digits after the binary point. This means that in this machine, $p = 5$.

Rounding up will be $u = 1.1010_2 \times 2^{-4}$. Rounding down will be $d = 1.1001_2 \times 2^{-4}$. To see which one is nearest, we can compute the midpoint of u and d.



0.1 is clearly bigger than the middle point, so we pick $1.1010_2 \times 2^{-4}$ with rounding to the nearest. We can write $\text{fl}(0.1) = 1.1010_2 \times 2^{-4}$ in this particular machine.

Question: In this same machine, meaning we still have $p = 5$ and we still use rounding to the nearest, what is $\text{fl}(1.10011_2 \times 2^{-4})$? (Hint: in binary, we break the tie by choosing the one whose least significant (rightmost) bit is 0.)

In the model where $\beta = 10, p = 5$, 1 is represented as 1.0000×10^0, the number that is closest to 1 (and bigger than 1) is 1.0001×10^0. There is a gap 10^{-4}. This gap is relative. The closest number to 1.0000×10^4 is 1.0001×10^4 (taking the bigger one again). This means we cannot represent any number between 10000 and 10001 with this machine. The gap (=1) is a lot bigger now, but the relative gap is still $1/10000 = 10^{-4}$.

In general, with base β and precision level p, this gap (between 1 and its closest number) is defined to be the *machine precision* or *machine epsilon* ϵ_m.[1] We have $\epsilon_m = \beta^{1-p}$. Machine epsilon is related to the *round-off error*, which is the error that results from replacing a number with its floating point form. The approximation x^* to x has *absolute error* $|x - x^*|$ and *relative error* $\dfrac{|x - x^*|}{|x|}$, provided that $x \neq 0$.

Example 1.2. Let us use base 10 with $p = 3$, in which case the machine precision is $\epsilon_m = 10^{-2}$. If we use the rounding to nearest method, 1.345 becomes 1.34 (both 1.34 and 1.35 are equally

[1] Some sources define the machine precision to be half of this gap.

close, we break the tie by choosing the number whose right most bit is smaller). Relative error $= \left|\frac{1.345 - 1.34}{1.345}\right| < \frac{0.005}{1} = 0.5 * \epsilon_m$ (The relative error is about 0.003.)

The number 1.345×10^4 will be rounded to 1.34×10^4. The absolute error is very big, but the relative error is $\left|\frac{1.345 \times 10^4 - 1.34 \times 10^4}{1.345 \times 10^4}\right| = \left|\frac{1.345 - 1.34}{1.345}\right|$, the same as before.

Example 1.3. Following Example 1.1, the absolute error in representing 0.1 is $|0.1 - \text{fl}(0.1)|$, which is less than half of the difference between u and d. The relative error is

$$\frac{|0.1 - \text{fl}(0.1)|}{0.1} < \frac{2^{-1}(u-d)}{d} = \frac{2^{-5} \cdot 2^{-4}}{1.1001_2 \times 2^{-4}} < 2^{-5}.$$

If we use rounding to the nearest, one has

$$\frac{|\text{fl}(x) - x|}{|x|} \leq 0.5\epsilon_m. \tag{1.3}$$

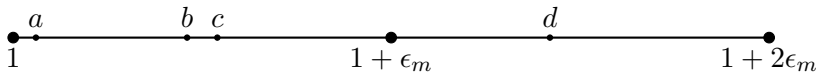
In other words, we may write

$$\text{fl}(x) = x(1+\epsilon), \quad \text{where } |\epsilon| < 0.5\epsilon_m.$$

The idea of machine epsilon is that

- Two numbers may be indistinguishable in a machine if their relative error is less than $0.5\epsilon_m$;
- Two numbers are definitely distinguishable in a machine if their relative error is more than ϵ_m.

Let's consider four numbers a, b, c, d as indicated below:



Using rounding to the nearest, $\text{fl}(a) = \text{fl}(b) = 1$ and a, b are within $0.5\epsilon_m$ relative error. $\text{fl}(b) \neq \text{fl}(c)$ even though they are within $0.5\epsilon_m$ relative error. $\text{fl}(c) = \text{fl}(d)$ even though they are almost ϵ_m apart. The only way to guarantee that two numbers have different representations is make sure they are relatively ϵ_m apart, like b and d.

Chapter 1 Exercises

1. Convert binary to decimal.
   (a) 110_2
   (b) 11.0101_2
2. Convert decimal to binary
   (a) 11.5
   (b) 1.3125
3. Find the **normalized** floating point representation

(a) 10.345, base 10, $p = 4$, rounding to the nearest.

(b) 295, base 10, $p = 2$, rounding to the nearest.

(c) 1.0101_2, base 2, $p = 3$, rounding up.

(d) 1.0101_2, base 2, $p = 3$, rounding down.

(e) 1.0101_2, base 2, $p = 3$, rounding to the nearest.

(f) 1.0101_2, base 2, $p = 4$, rounding to the nearest.

4. Let us use base 10 with $p = 2$. If we use the rounding to the nearest method, what is the relative error of 1.23? What is the relative error of 10.61?

5. Following Exercise 3(f), what is the machine epsilon? what is the relative error in rounding this number? Does the rule (1.3) check out?

6. Directly add in binary (No rounding involved).

   (a) $0.111_2 + 11.101_2$

   (b) $1.1010_2 \times 2^{-4} + 1.1010_2 \times 2^{-3}$.

7. What is the smallest and biggest number that can be represented in IEEE standard using double precision?

8. Compute the machine result of 0.1+0.2 if we retain 2 digits after the binary point ($p = 3$).

9. This problem is from Chapter 5 problem 15 of [2].

   In the 1991 Gulf War, the Patriot misslie defense system failed due to roundoff error. The troubles stemmed from a computer that performed the tracking calculations with an internal clock whose integer values in tenths of a second were converted to seconds by multiplying by a 24-bit binary approximation to 0.1:

   $$0.1 \approx 0.00011001100110011001100_2$$

   (a) Convert the binary number above to a fraction. Call it x.

   (b) Compute the absolute difference between 0.1 and x.

   (c) What is the time error in seconds after 100 hours of operation (i.e., the value of $|360,000 - 3,600,000x|$)?

   On February 25, 1991, a Patriot battery system, which was to protet the Dhahran Air Base, had been operating for over 100 consecutive hours. The roundoff error caused the system not to track an incoming Scud missle, which slipped through the defense system and detonated on US Army barracks, killing 28 American soldiers.

10. Find the most accurate approximation of the roots of $x^2 - 75.7x + 1$ with $p = 4$. Compute the relative errors for both roots.

11. Given $p(x) = 1 + 2x + 3x^2 + 3x^3 + 4x^4$,

    (a) Find the nested form of $p(x)$.

    (b) How many flops are needed to evaluate this polynomial at one point, in the original form, and in the nested form?

Chapter 1 Python Exercises

Bibliography

[1] Richard L. Burden, and J. Douglas Faires. *Numerical analysis.* Cengage Learning, 9th (2010).

[2] Anne, Greenbaum, and Timothy P. Chartier. *Numerical methods: design, analysis, and computer implementation of algorithms.* Princeton University Press, 2012.

[3] Ab Mooijaart, Matthijs Warrens, and Eeke van der Burg, *An introduction to matrix algebra with Matlab*, manuscript, 2006

[4] Lloyd N. Trefethen, and David Bau III. *Numerical linear algebra.* Vol. 50. Siam, 1997.