Chapter 4: Roundoff and Truncation Errors

Department of Mechanical Engineering
Choi Hae Jin

Chapter Objectives

❑ Roundoff Error

- Understanding how roundoff errors occur because digital computers have a limited ability to represent numbers.
- Understanding why floating-point numbers have limits on their range and precision.

❑ Truncation Error

- Recognizing that truncation errors occur when exact mathematical formulations are represented by approximations.
- Knowing how to use the Taylor series to estimate truncation errors.
- Understanding how to write forward, backward, and centered finite-difference approximations of the first and second derivatives.
- Recognizing that efforts to minimize truncation errors can sometimes increase roundoff errors.

Error Definitions

- **True error (E_t): the difference between the true value and the approximation.**
  - E_t = True value – approximation

- **Absolute error ($|E_t|$): the absolute difference between the true value and the approximation.**

- **True fractional relative error: the true error divided by the true value.**
  - True fractional relative error = (true value – approximation)/true value

- **Relative error (ε_t): the true fractional relative error expressed as a percentage.**
  - ε_t= true fractional relative error * 100%

Error Definitions (cont)

- The previous definitions of error relied on knowing a true value. If that is not the case, approximations can be made to the error.
- The **approximate percent relative error** can be given as the approximate error divided by the approximation, expressed as a percentage - though this presents the challenge of finding the approximate error!
- For iterative processes, the error can be approximated as **the difference in values between successive iterations**.

Using Error Estimates

- Often, when performing calculations, we may not be concerned with the sign of the error but are interested in whether the absolute value of the percent relative error is lower than a prespecified tolerance ε_s. For such cases, the computation is repeated until $|\varepsilon_a| < \varepsilon_s$
- This relationship is referred to as a ***stopping criterion***.

Example 4.1 (1)

- Q. How many terms are required in calculation of $e^{0.5}$(=1.648721...) using a Maclaurin series expansion, in which the result is correct to at least 3 significant figure?

$$e^{x} = 1 + x + \frac{x^{2}}{2!} + \frac{x^{3}}{3!} + \cdots + \frac{x^{n}}{n!}$$

Maclaurin series

Error criterion for 3 significant figure

$$\varepsilon_{s} = (0.5 \times 10^{2-n})\% = (0.5 \times 10^{2-3})\% = 0.05\%$$

(Scarborough, 1966)

Example 4.1 (2)

Terms	Results	ε_t (%)	ε_a (%)
1	1	39.3	
2	1.5	9.02	33.3
3	1.625	1.44	7.69
4	1.645800000	0.175	1.27
5	1.648437500	0.0172	0.158
6	1.648697917	0.00142	0.0158

Scarborough Error Criterion is Conservative!!

Roundoff Errors

- ***Roundoff errors*** **arise because digital computers cannot represent some quantities exactly. There are two major facets of roundoff errors involved in numerical calculations:**
  - Digital computers have size and precision limits on their ability to represent numbers.
  - Certain numerical manipulations are highly sensitive to roundoff errors.

Computer Number Representation

- Bit : binary number (0/1)
- Byte : 8 bit
- Word
  - Basic unit for expressing number
  - ex) 16 bit or 2byte word
- Decimal expression (positional notation)

$$8642.9 = (8 \times 10^3) + (6 \times 10^2) + (4 \times 10^1) + (2 \times 10^0) + (9 \times 10^{-1})$$

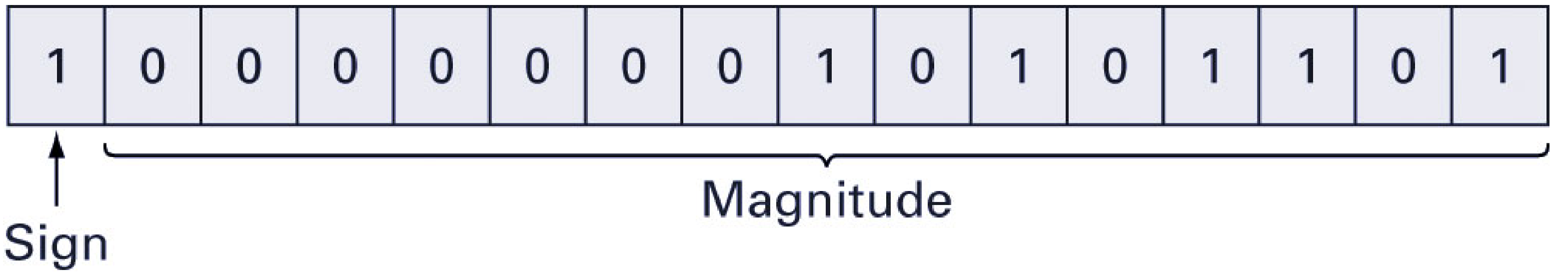
- Binary expression (positional notation)

$$101.1 = (1 \times 2^2) + (0 \times 2^1) + (1 \times 2^0) + (1 \times 2^{-1}) = 4 + 0 + 1 + 0.5 = 5.5$$

Integer Representation

- **For an n bit word, the range would be from $-2^{n-1} + 2^{n-1}-1$**
- **The numbers above or below the range can't be represented**

Ex. 16 bit word



$$(10101101)_2 = 2^7 + 2^5 + 2^3 + 2^2 + 2^0 = 128 + 32 + 8 + 4 + 1$$
$$= (173)_{10}$$

Integer Representation

- Upper limit, Lower limit and zero for 16 bit word

$$(0111\cdots111)_2 = 2^{14} + 2^{13} + \bullet\bullet\bullet + 2^2 + 2^1 + 2^0 = 32,767 = 2^{15} - 1$$

$$(0000\cdots000)_2 = 0$$

$$(1111\cdots111)_2 = 2^{14} + 2^{13} + \bullet\bullet\bullet + 2^2 + 2^1 + 2^0 = -32,767 = -(2^{15} - 1)$$

$$(1000\cdots000)_2 = -32,768$$

-32768 (-2^{n-1})< integer < 32767 (2^{n-1}-1)

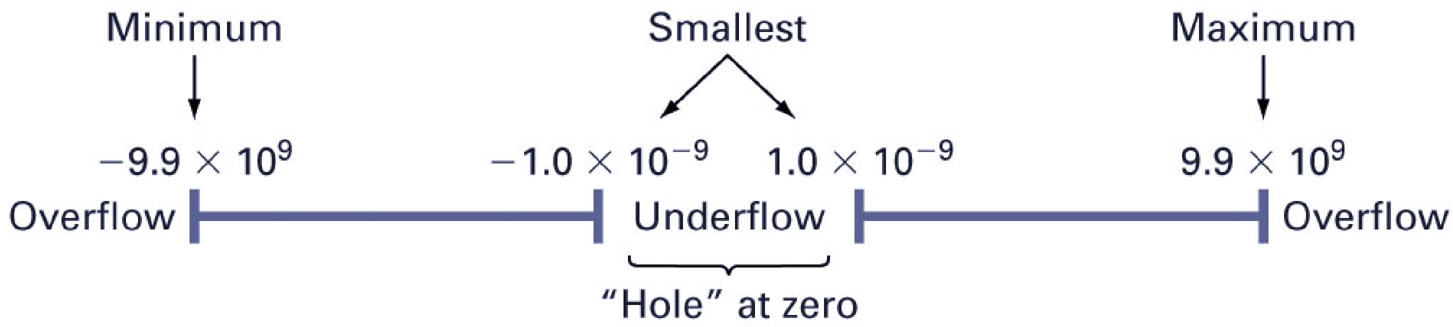
Floating Point Representation

- The number is expressed as $s \times b^e$
  where, s: the mantissa (significand), b:base, e: exponent

- Ex.) Base-10 computer with a 5 bit word

$$S_1 d_1 . d_2 \times 10^{S_0 d_0}$$

Range = $+9.9 \times 10^{+9} \sim +1.0 \times 10^{-9}$



Roundoff Errors

- Base-10 computer with a 5 bit word

$$S_1 d_1 . d_2 \times 10^{S_0 d_0}$$

- 2^{-5}= 0.03125 → 3.1 x 10^{-2}

→ roundoff error =

(0.03125-0.031)/0.03125 = 0.008 = 0.8%

- Because of the limited number of bits for significand and exponent, Roundoff errors is occur.

  π= 3.141593for 16-bit word computer
  π= 3.14159265358979 for 32-bit word computer

- Although adding significand digits can improve the approximation, such quantities will always have some roundoff error when stored in a computer

Computer Number Representation

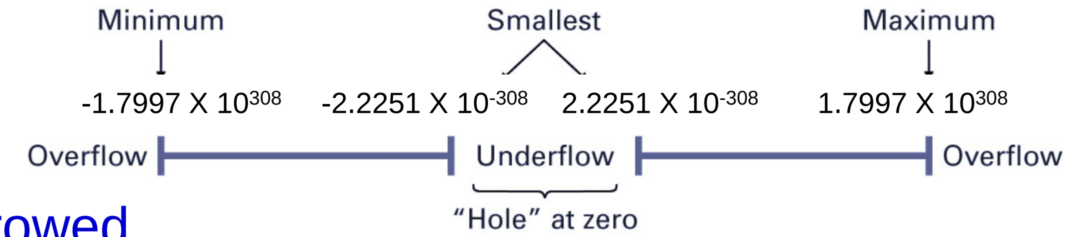
- By default, MATLAB has adopted the IEEE double-precision format in which eight bytes (64 bits) are used to represent floating-point numbers: $n=\pm(1+f) \times 2^{e}$
- The sign is determined by a sign bit
- The mantissa f is determined by a 52-bit binary number
- The exponent e is determined by an 11-bit binary number, from which 1023 is subtracted to get e

Floating Point Ranges

- The exponent range is -1022 to 1023.
  (11 bits including 1 bit for sign)

- The largest possible number MATLAB can store has
  - $+1.111111\ldots111 \text{ X } 2^{1023} = (2-2^{-52}) \text{X } 2^{1023}$
  - This yields approximately $2^{1024} = 1.7997 \text{ X } 10^{308}$

- The smallest possible number MATLAB can store with full precision has
  - $+1.00000\ldots00000 \text{ X } 2^{-1022}$
  - This yields $2^{-1022} = 2.2251 \text{ X } 10^{-308}$



Note: Hole was greatly narrowed.

Maximum, Minimum & Machine epsilon in MATLAB

- The 52 bits for the significand ***f*** correspond to about 15 to 16 base-10 digits.
- The machine epsilon in MATLAB's representation of a number is thus $2^{-52}=2.2204 \times 10^{-16}$

Command Window

File Edit View Web Window Help

```
>> format long
>> realmax
ans =
   1.797693134862316e+308
>> realmin
ans =
   2.225073858507201e-308
>> eps              (machine epsilon)
ans =
   2.220446049250313e-016
```

Numerical Problems

- $1.557+0.04341 = 0.1557 \times 10^1 + 0.004341 \times 10^1$
  $= 0.160041 \times 10^1 = 0.1600 \times 10^1$
- The excess number of digits were chopped off, leading to error.

- $36.41 - 26.86 = 0.3641 \times 10^2 - 0.3641 \times 10^2$
  $= 0.0955 \times 10^2 \rightarrow 0.9550 \times 10^1$
- The zero added to the end.

- $0.7642 \times 10^3 - 0.7641 \times 10^3 = 0.0001 \times 10^3 = 0.1000$
- Three zeros are appended.

Truncation Errors

- ***Truncation errors* are those that result from using an approximation in place of an exact mathematical procedure.**
- **Example 1: approximation to a derivative using a finite-difference equation:**

$$\frac{dv}{dt} \cong \frac{\Delta v}{\Delta t} = \frac{v(t_{i+1}) - v(t_i)}{t_{i+1} - t_i}$$

Example 2: The Taylor Series

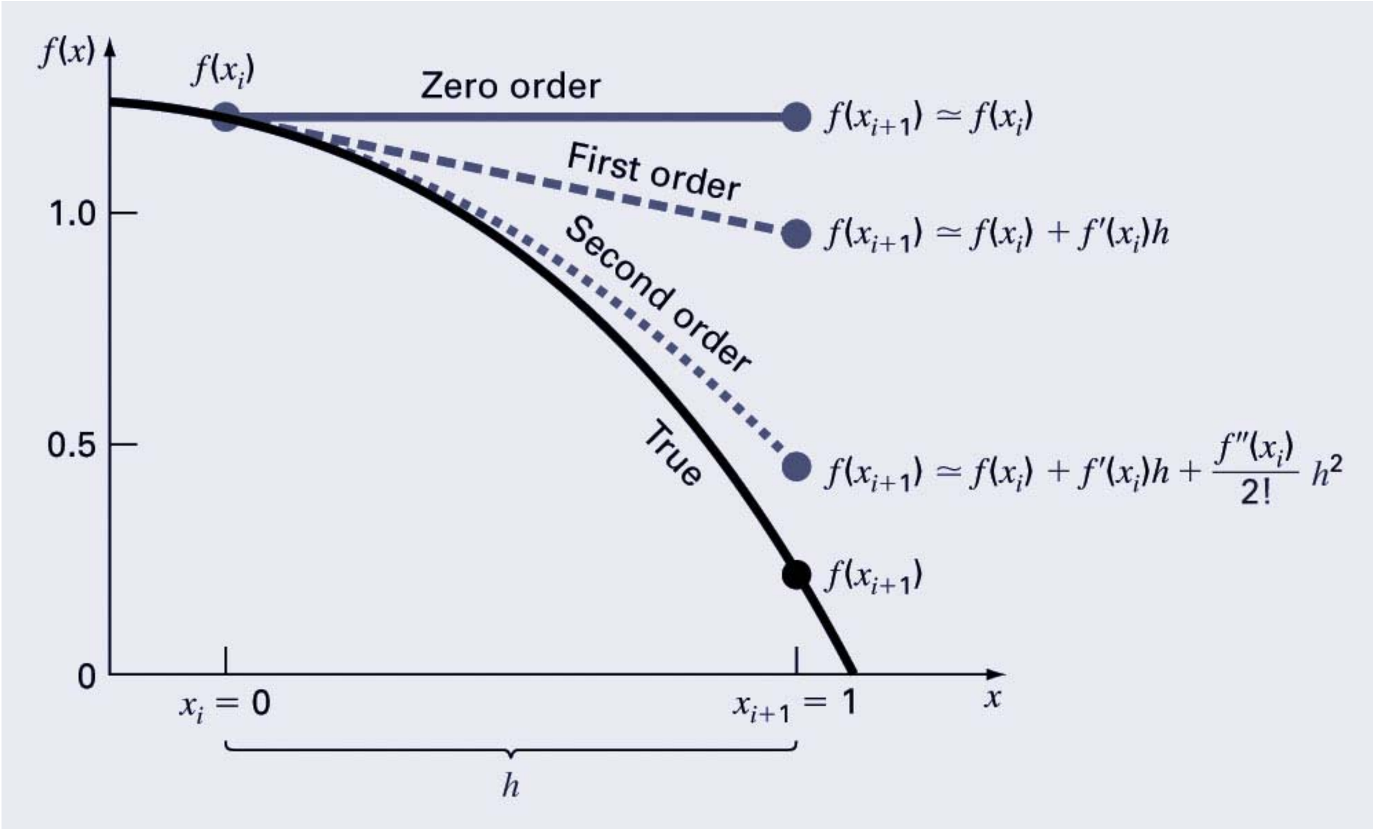
The Taylor Theorem and Series

- The *Taylor theorem* states that any smooth function can be approximated as a polynomial.
- The *Taylor series* provides a means to express this idea mathematically.

$$f(x) = f(x_0) + \frac{x - x_0}{1!} f'(x_0) + \frac{(x - x_0)^2}{2!} f''(x_0) + \cdots + \frac{(x - x_0)^n}{n!} f^{(n)}(x_0) + R_n$$

The Taylor Series

$$f(x_{i+1}) = f(x_i) + f'(x_i)h + \frac{f''(x_i)}{2!}h^2 + \frac{f^{(3)}(x_i)}{3!}h^3 + \cdots + \frac{f^{(n)}(x_i)}{n!}h^n + R_n$$



Truncation Error

- **In general, the *n*th order Taylor series expansion will be exact for an *n*th order polynomial.**
- **In other cases, the remainder term R_n is of the order of h^{n+1}, meaning:**
  - The more terms are used, the smaller the error, and
  - The smaller the spacing, the smaller the error for a given number of terms.

Numerical Differentiation

- The first order Taylor series can be used to calculate approximations to derivatives:
  - Given: $f(x_{i+1}) = f(x_i) + f^{'}(x_i)h + O(h^2)$
  - Then: $f^{'}(x_i) = \dfrac{f(x_{i+1}) - f(x_i)}{h} + O(h)$
- This is termed a "forward" difference because it utilizes data at *i* and *i*+1 to estimate the derivative.

Differentiation (cont)

- There are also backward difference and centered difference approximations, depending on the points used:
- **Forward:**

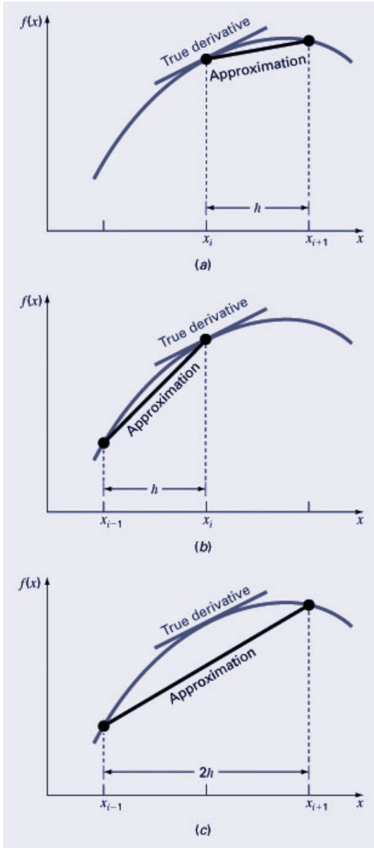
$$f'(x_i) = \frac{f(x_{i+1}) - f(x_i)}{h} + O(h)$$

- **Backward:**

$$f'(x_i) = \frac{f(x_i) - f(x_{i-1})}{h} + O(h)$$

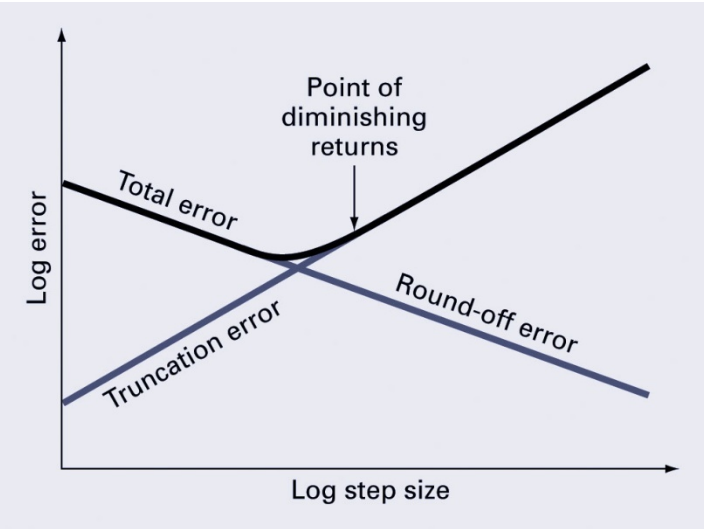
- **Centered:**

$$f'(x_i) = \frac{f(x_{i+1}) - f(x_{i-1})}{2h} + O(h^2)$$



Total Numerical Error

- **The *total numerical error* is the summation of the truncation and roundoff errors.**
- **The truncation error generally *increases* as the step size increases, while the roundoff error *decreases* as the step size increases - this leads to a point of diminishing returns for step size.**



Other Errors

- **Blunders - errors caused by malfunctions of the computer or human imperfection.**
- **Model errors - errors resulting from incomplete mathematical models.**
- **Data uncertainty - errors resulting from the accuracy and/or precision of the data.**