Mathematical Preliminaries and Error Analysis

Numerical Analysis I

Wei-Cheng Wang [1]

Department of Mathematics
National TsingHua University

Fall 2010

[1] These lecture slides are based on Prof. Tsung-Ming Huang(NTNU)'s original slides

Outline

1. **Round-off errors and computer arithmetic**
   - IEEE standard floating-point format
   - Absolute and Relative Errors
   - Machine Epsilon
   - Loss of Significance

2. **Algorithms and Convergence**
   - Algorithm
   - Stability
   - Rate of convergence

Outline

1. **Round-off errors and computer arithmetic**
   - IEEE standard floating-point format
   - Absolute and Relative Errors
   - Machine Epsilon
   - Loss of Significance

2. **Algorithms and Convergence**
   - Algorithm
   - Stability
   - Rate of convergence

Terminologies

- binary: 二進位 , decimal: 十進位 , hexadecimal: 十六進位
- exponent: 指數, mantissa: 尾數
- floating point numbers: 浮點數
- chopping: 無條件捨去, rounding: 四捨五入(X捨Y入)
- single precision: 單精度, double precisiom: 雙精度
- roundoff error: 捨入誤差
- significant digits: 有效位數
- loss of significance: 有效位數喪失

Example

What is the binary representation of $\frac{2}{3}$?

Solution: To determine the binary representation for $\frac{2}{3}$, we write

$$\frac{2}{3} = (0.a_1 a_2 a_3 \ldots)_2.$$

Multiply by 2 to obtain

$$\frac{4}{3} = (a_1.a_2 a_3 \ldots)_2.$$

Therefore, we get $a_1 = 1$ by taking the integer part of both sides.

Example

What is the binary representation of $\frac{2}{3}$?

Solution: To determine the binary representation for $\frac{2}{3}$, we write

$$\frac{2}{3} = (0.a_1a_2a_3\ldots)_2.$$

Multiply by 2 to obtain

$$\frac{4}{3} = (a_1.a_2a_3\ldots)_2.$$

Therefore, we get $a_1 = 1$ by taking the integer part of both sides.

Example

What is the binary representation of $\frac{2}{3}$?

Solution: To determine the binary representation for $\frac{2}{3}$, we write

$$\frac{2}{3} = (0.a_1 a_2 a_3 \ldots)_2.$$

Multiply by 2 to obtain

$$\frac{4}{3} = (a_1.a_2 a_3 \ldots)_2.$$

Therefore, we get $a_1 = 1$ by taking the integer part of both sides.

Subtracting 1, we have

$$\frac{1}{3} = (0.a_2 a_3 a_4 \ldots)_2.$$

Repeating the previous step, we arrive at

$$\frac{2}{3} = (0.101010\ldots)_2.$$

Subtracting 1, we have

$$\frac{1}{3} = (0.a_2a_3a_4\ldots)_2.$$

Repeating the previous step, we arrive at

$$\frac{2}{3} = (0.101010\ldots)_2.$$

∎

- In the computational world, each representable number has only a fixed and finite number of digits.
- For any real number x, let

$$x = \pm 1.a_1a_2 \cdots a_ta_{t+1}a_{t+2} \cdots \times 2^m,$$

denote the normalized scientific binary representation of x.
- In 1985, the IEEE (Institute for Electrical and Electronic Engineers) published a report called *Binary Floating Point Arithmetic Standard 754-1985*. In this report, formats were specified for single, double, and extended precisions, and these standards are generally followed by microcomputer manufactures to design floating-point hardware.

IEEE standard floating-point format

- In the computational world, each representable number has only a fixed and finite number of digits.
- For any real number x, let

$$x = \pm 1.a_1 a_2 \cdots a_t a_{t+1} a_{t+2} \cdots \times 2^m,$$

denote the normalized scientific binary representation of x.

- In 1985, the IEEE (Institute for Electrical and Electronic Engineers) published a report called *Binary Floating Point Arithmetic Standard 754-1985*. In this report, formats were specified for single, double, and extended precisions, and these standards are generally followed by microcomputer manufactures to design floating-point hardware.

- In the computational world, each representable number has only a fixed and finite number of digits.
- For any real number x, let

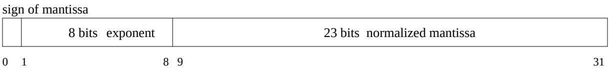
$$x = \pm 1.a_1a_2\cdots a_ta_{t+1}a_{t+2}\cdots \times 2^m,$$

denote the normalized scientific binary representation of x.

- In 1985, the IEEE (Institute for Electrical and Electronic Engineers) published a report called *Binary Floating Point Arithmetic Standard 754-1985*. In this report, formats were specified for single, double, and extended precisions, and these standards are generally followed by microcomputer manufactures to design floating-point hardware.

Single precision

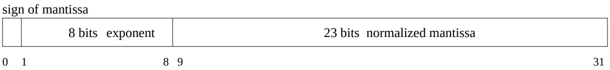
- The single precision IEEE standard floating-point format allocates 32 bits for the normalized floating-point number $\pm q \times 2^m$ as shown in the following figure.



- The first bit is a sign indicator, denoted s. This is followed by an 8-bit exponent c and a 23-bit mantissa f.
- The base for the exponent and mantissa is 2, and the actual exponent is $c - 127$. The value of c is restricted by the inequality $0 \leq c \leq 255$.

Single precision

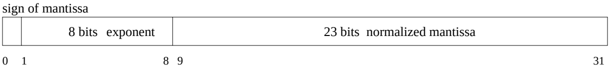
- The single precision IEEE standard floating-point format allocates 32 bits for the normalized floating-point number $\pm q \times 2^m$ as shown in the following figure.



- The first bit is a sign indicator, denoted s. This is followed by an 8-bit exponent c and a 23-bit mantissa f.
- The base for the exponent and mantissa is 2, and the actual exponent is $c - 127$. The value of c is restricted by the inequality $0 \leq c \leq 255$.

Single precision

- The single precision IEEE standard floating-point format allocates 32 bits for the normalized floating-point number $\pm q \times 2^m$ as shown in the following figure.



- The first bit is a sign indicator, denoted s. This is followed by an 8-bit exponent c and a 23-bit mantissa f.
- The base for the exponent and mantissa is 2, and the actual exponent is $c - 127$. The value of c is restricted by the inequality $0 \leq c \leq 255$.

IEEE standard floating-point format

- The actual exponent of the number is restricted by the inequality $-127 \leq c - 127 \leq 128$.
- A normalization is imposed that requires that the leading digit in fraction be 1, and this digit is not stored as part of the 23-bit mantissa.
- Using this system gives a floating-point number of the form

$$(-1)^s 2^{c-127}(1+f).$$

IEEE standard floating-point format

- The actual exponent of the number is restricted by the inequality $-127 \le c - 127 \le 128$.
- A normalization is imposed that requires that the leading digit in fraction be 1, and this digit is not stored as part of the 23-bit mantissa.
- Using this system gives a floating-point number of the form

$$(-1)^s 2^{c-127}(1+f).$$

IEEE standard floating-point format

- The actual exponent of the number is restricted by the inequality $-127 \leq c - 127 \leq 128$.
- A normalization is imposed that requires that the leading digit in fraction be 1, and this digit is not stored as part of the 23-bit mantissa.
- Using this system gives a floating-point number of the form

$$(-1)^s 2^{c-127}(1+f).$$

Example

What is the decimal number of the machine number

$$0\underline{10000001}01000000000000000000000?$$

1. The leftmost bit is zero, which indicates that the number is positive.
2. The next 8 bits, 10000001, are equivalent to
$$c = 1 \cdot 2^7 + 0 \cdot 2^6 + \cdots + 0 \cdot 2^1 + 1 \cdot 2^0 = 129.$$
The exponential part of the number is $2^{129-127} = 2^2$.
3. The final 23 bits specify that the mantissa is
$$f = 0 \cdot (2)^{-1} + 1 \cdot (2)^{-2} + 0 \cdot (2)^{-3} + \cdots + 0 \cdot (2)^{-23} = 0.25.$$
4. Consequently, this machine number precisely represents the decimal number
$$(-1)^s 2^{c-127}(1+f) = 2^2 \cdot (1 + 0.25) = 5.$$

Example

What is the decimal number of the machine number

$$0\underline{10000001}01000000000000000000000?$$

1. The leftmost bit is zero, which indicates that the number is positive.
2. The next 8 bits, 10000001, are equivalent to

   $$c = 1 \cdot 2^7 + 0 \cdot 2^6 + \cdots + 0 \cdot 2^1 + 1 \cdot 2^0 = 129.$$

   The exponential part of the number is $2^{129-127} = 2^2$.
3. The final 23 bits specify that the mantissa is

   $$f = 0 \cdot (2)^{-1} + 1 \cdot (2)^{-2} + 0 \cdot (2)^{-3} + \cdots + 0 \cdot (2)^{-23} = 0.25.$$
4. Consequently, this machine number precisely represents the decimal number

   $$(-1)^s 2^{c-127}(1+f) = 2^2 \cdot (1+0.25) = 5.$$

Example

What is the decimal number of the machine number

$$0\underline{10000001}01000000000000000000000?$$

1. The leftmost bit is zero, which indicates that the number is positive.
2. The next 8 bits, 10000001, are equivalent to

$$c = 1 \cdot 2^7 + 0 \cdot 2^6 + \cdots + 0 \cdot 2^1 + 1 \cdot 2^0 = 129.$$

   The exponential part of the number is $2^{129-127} = 2^2$.
3. The final 23 bits specify that the mantissa is

$$f = 0 \cdot (2)^{-1} + 1 \cdot (2)^{-2} + 0 \cdot (2)^{-3} + \cdots + 0 \cdot (2)^{-23} = 0.25.$$

4. Consequently, this machine number precisely represents the decimal number

$$(-1)^s 2^{c-127}(1+f) = 2^2 \cdot (1 + 0.25) = 5.$$

Example

What is the decimal number of the machine number

$$0\underline{10000001}01000000000000000000000?$$

1. The leftmost bit is zero, which indicates that the number is positive.
2. The next 8 bits, 10000001, are equivalent to
$$c = 1 \cdot 2^7 + 0 \cdot 2^6 + \cdots + 0 \cdot 2^1 + 1 \cdot 2^0 = 129.$$
The exponential part of the number is $2^{129-127} = 2^2$.
3. The final 23 bits specify that the mantissa is
$$f = 0 \cdot (2)^{-1} + 1 \cdot (2)^{-2} + 0 \cdot (2)^{-3} + \cdots + 0 \cdot (2)^{-23} = 0.25.$$
4. Consequently, this machine number precisely represents the decimal number
$$(-1)^s 2^{c-127}(1+f) = 2^2 \cdot (1 + 0.25) = 5.$$

Example

What is the decimal number of the machine number

$$0\underline{10000001}01000000000000000000000?$$

1. The leftmost bit is zero, which indicates that the number is positive.
2. The next 8 bits, 10000001, are equivalent to

$$c = 1 \cdot 2^7 + 0 \cdot 2^6 + \cdots + 0 \cdot 2^1 + 1 \cdot 2^0 = 129.$$

   The exponential part of the number is $2^{129-127} = 2^2$.
3. The final 23 bits specify that the mantissa is

$$f = 0 \cdot (2)^{-1} + 1 \cdot (2)^{-2} + 0 \cdot (2)^{-3} + \cdots + 0 \cdot (2)^{-23} = 0.25.$$

4. Consequently, this machine number precisely represents the decimal number

$$(-1)^s 2^{c-127}(1+f) = 2^2 \cdot (1 + 0.25) = 5.$$

Example

What is the decimal number of the machine number

$$0\underline{10000001}00111111111111111111111?$$

1. The final 23 bits specify that the mantissa is

$$\begin{aligned} f &= 0\cdot(2)^{-1} + 0\cdot(2)^{-2} + 1\cdot(2)^{-3} + \cdots + 1\cdot(2)^{-23} \\ &= 0.2499998807907105. \end{aligned}$$

2. Consequently, this machine number precisely represents the decimal number

$$\begin{aligned} (-1)^s 2^{c-127}(1+f) &= 2^2\cdot(1+0.2499998807907105) \\ &= 4.999999523162842. \end{aligned}$$

Example

What is the decimal number of the machine number

$$0\underline{10000001}00111111111111111111111?$$

1. The final 23 bits specify that the mantissa is

$$\begin{aligned} f &= 0\cdot(2)^{-1} + 0\cdot(2)^{-2} + 1\cdot(2)^{-3} + \cdots + 1\cdot(2)^{-23} \\ &= 0.2499998807907105. \end{aligned}$$

2. Consequently, this machine number precisely represents the decimal number

$$\begin{aligned} (-1)^s 2^{c-127}(1+f) &= 2^2\cdot(1+0.2499998807907105) \\ &= 4.999999523162842. \end{aligned}$$

Example

What is the decimal number of the machine number

$$0\underline{10000001}00111111111111111111111?$$

1. The final 23 bits specify that the mantissa is

$$\begin{aligned} f &= 0\cdot(2)^{-1}+0\cdot(2)^{-2}+1\cdot(2)^{-3}+\cdots+1\cdot(2)^{-23} \\ &= 0.2499998807907105. \end{aligned}$$

2. Consequently, this machine number precisely represents the decimal number

$$\begin{aligned} (-1)^s 2^{c-127}(1+f) &= 2^2\cdot(1+0.2499998807907105) \\ &= 4.999999523162842. \end{aligned}$$

Example

What is the decimal number of the machine number

$$0\underline{10000001}01000000000000000000001?$$

1. The final 23 bits specify that the mantissa is

$$\begin{aligned} f &= 0\cdot 2^{-1} + 1\cdot 2^{-2} + 0\cdot 2^{-3} + \cdots + 0\cdot 2^{-22} + 1\cdot 2^{-23} \\ &= 0.2500001192092896. \end{aligned}$$

2. Consequently, this machine number precisely represents the decimal number

$$\begin{aligned} (-1)^s 2^{c-127}(1+f) &= 2^2\cdot(1+0.2500001192092896) \\ &= 5.000000476837158. \end{aligned}$$

Example

What is the decimal number of the machine number

$$0\underline{10000001}01000000000000000000001?$$

1. The final 23 bits specify that the mantissa is

$$\begin{aligned} f &= 0 \cdot 2^{-1} + 1 \cdot 2^{-2} + 0 \cdot 2^{-3} + \cdots + 0 \cdot 2^{-22} + 1 \cdot 2^{-23} \\ &= 0.2500001192092896. \end{aligned}$$

2. Consequently, this machine number precisely represents the decimal number

$$\begin{aligned} (-1)^s 2^{c-127}(1+f) &= 2^2 \cdot (1 + 0.2500001192092896) \\ &= 5.000000476837158. \end{aligned}$$

Example

What is the decimal number of the machine number

$$0\underline{10000001}01000000000000000000001?$$

1. The final 23 bits specify that the mantissa is

$$\begin{aligned} f &= 0 \cdot 2^{-1} + 1 \cdot 2^{-2} + 0 \cdot 2^{-3} + \cdots + 0 \cdot 2^{-22} + 1 \cdot 2^{-23} \\ &= 0.2500001192092896. \end{aligned}$$

2. Consequently, this machine number precisely represents the decimal number

$$\begin{aligned} (-1)^s 2^{c-127}(1+f) &= 2^2 \cdot (1 + 0.2500001192092896) \\ &= 5.000000476837158. \end{aligned}$$

Summary

Above three examples

$$
\begin{array}{lcl}
0\underline{10000001}00111111111111111111111 & \Rightarrow & 4.999999523162842 \\
0\underline{10000001}01000000000000000000000 & \Rightarrow & 5 \\
0\underline{10000001}01000000000000000000001 & \Rightarrow & 5.000000476837158
\end{array}
$$

- Only a relatively small subset of the real number system is used for the representation of all the real numbers.
- This subset, which are called the *floating-point numbers*, contains only rational numbers, both positive and negative.
- When a number can not be represented exactly with the fixed finite number of digits in a computer, a near-by floating-point number is chosen for approximate representation.

Summary

Above three examples

$$
\begin{array}{lcl}
0\underline{10000001}00111111111111111111111 & \Rightarrow & 4.99999523162842 \\
0\underline{10000001}01000000000000000000000 & \Rightarrow & 5 \\
0\underline{10000001}01000000000000000000001 & \Rightarrow & 5.00000476837158
\end{array}
$$

- Only a relatively small subset of the real number system is used for the representation of all the real numbers.
- This subset, which are called the *floating-point numbers*, contains only rational numbers, both positive and negative.
- When a number can not be represented exactly with the fixed finite number of digits in a computer, a near-by floating-point number is chosen for approximate representation.

Summary

Above three examples

$$
\begin{array}{lcl}
0\underline{10000001}00111111111111111111111 & \Rightarrow & 4.99999523162842 \\
0\underline{10000001}01000000000000000000000 & \Rightarrow & 5 \\
0\underline{10000001}01000000000000000000001 & \Rightarrow & 5.00000476837158
\end{array}
$$

- Only a relatively small subset of the real number system is used for the representation of all the real numbers.
- This subset, which are called the *floating-point numbers*, contains only rational numbers, both positive and negative.
- When a number can not be represented exactly with the fixed finite number of digits in a computer, a near-by floating-point number is chosen for approximate representation.

Summary

Above three examples

$$
\begin{array}{lcl}
0\underline{10000001}00111111111111111111111 & \Rightarrow & 4.99999952316284 \\
0\underline{10000001}01000000000000000000000 & \Rightarrow & 5 \\
0\underline{10000001}01000000000000000000001 & \Rightarrow & 5.00000047683716
\end{array}
$$

- Only a relatively small subset of the real number system is used for the representation of all the real numbers.
- This subset, which are called the *floating-point numbers*, contains only rational numbers, both positive and negative.
- When a number can not be represented exactly with the fixed finite number of digits in a computer, a near-by floating-point number is chosen for approximate representation.

The smallest (normalized) positive number

Let $s = 0$, $c = 1$ and $f = 0$. This cooresponds to

$$2^{-126} \cdot (1 + 0) \approx 1.175 \times 10^{-38}$$

The largest number

Let $s = 0$, $c = 254$ and $f = 1 - 2^{-23}$ which is equivalent to

$$2^{127} \cdot (2 - 2^{-23}) \approx 3.403 \times 10^{38}$$

Definition

If a number x with $|x| < 2^{-126} \cdot (1 + 0)$, then we say that an *underflow* has occurred and is generally set to zero. It is sometimes referred to as an IEEE 'subnormal' or 'denormal' and corresponds to $c = 0$. If $|x| > 2^{127} \cdot (2 - 2^{-23})$, then we say that an *overflow* has occurred.

The smallest (normalized) positive number

Let $s = 0$, $c = 1$ and $f = 0$. This cooresponds to

$$2^{-126} \cdot (1 + 0) \approx 1.175 \times 10^{-38}$$

The largest number

Let $s = 0$, $c = 254$ and $f = 1 - 2^{-23}$ which is equivalent to

$$2^{127} \cdot (2 - 2^{-23}) \approx 3.403 \times 10^{38}$$

Definition

If a number x with $|x| < 2^{-126} \cdot (1 + 0)$, then we say that an *underflow* has occurred and is generally set to zero. It is sometimes referred to as an IEEE 'subnormal' or 'denormal' and corresponds to $c = 0$. If $|x| > 2^{127} \cdot (2 - 2^{-23})$, then we say that an *overflow* has occurred.

The smallest (normalized) positive number

Let $s = 0$, $c = 1$ and $f = 0$. This cooresponds to

$$2^{-126} \cdot (1 + 0) \approx 1.175 \times 10^{-38}$$

The largest number

Let $s = 0$, $c = 254$ and $f = 1 - 2^{-23}$ which is equivalent to

$$2^{127} \cdot (2 - 2^{-23}) \approx 3.403 \times 10^{38}$$

Definition

If a number x with $|x| < 2^{-126} \cdot (1 + 0)$, then we say that an *underflow* has occurred and is generally set to zero. It is sometimes referred to as an IEEE 'subnormal' or 'denormal' and corresponds to $c = 0$. If $|x| > 2^{127} \cdot (2 - 2^{-23})$, then we say that an *overflow* has occurred.

The smallest (normalized) positive number

Let $s = 0$, $c = 1$ and $f = 0$. This cooresponds to

$$2^{-126} \cdot (1 + 0) \approx 1.175 \times 10^{-38}$$

The largest number

Let $s = 0$, $c = 254$ and $f = 1 - 2^{-23}$ which is equivalent to

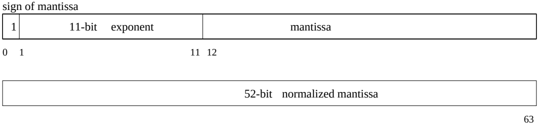
$$2^{127} \cdot (2 - 2^{-23}) \approx 3.403 \times 10^{38}$$

Definition

If a number x with $|x| < 2^{-126} \cdot (1 + 0)$, then we say that an *underflow* has occurred and is generally set to zero. It is sometimes referred to as an IEEE 'subnormal' or 'denormal' and corresponds to $c = 0$. If $|x| > 2^{127} \cdot (2 - 2^{-23})$, then we say that an *overflow* has occurred.

Double precision

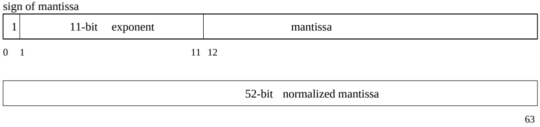
- A floating point number in double precision IEEE standard format uses two words (64 bits) to store the number as shown in the following figure.



- The first bit is a sign indicator, denoted s. This is followed by an 11-bit exponent c and a 52-bit mantissa f.
- The actual exponent is $c - 1023$.

Double precision

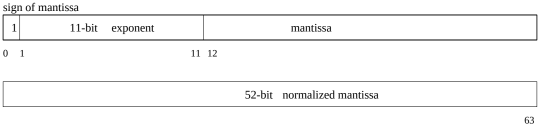
- A floating point number in double precision IEEE standard format uses two words (64 bits) to store the number as shown in the following figure.



- The first bit is a sign indicator, denoted s. This is followed by an 11-bit exponent c and a 52-bit mantissa f.
- The actual exponent is $c - 1023$.

Double precision

- A floating point number in double precision IEEE standard format uses two words (64 bits) to store the number as shown in the following figure.



- The first bit is a sign indicator, denoted s. This is followed by an 11-bit exponent c and a 52-bit mantissa f.
- The actual exponent is $c - 1023$.

Format of floating-point number

$$(-1)^s \times (1+f) \times 2^{c-1023}$$

The smallest (normalized) positive number

Let $s=0$, $c=1$ and $f=0$ which is equivalent to

$$2^{-1022}\cdot(1+0)\approx 2.225\times 10^{-308}.$$

The largest number

Let $s=0$, $c=2046$ and $f=1-2^{-52}$ which is equivalent to

$$2^{1023}\cdot(2-2^{-52})\approx 1.798\times 10^{308}.$$

IEEE standard floating-point format

Format of floating-point number

$$(-1)^s \times (1+f) \times 2^{c-1023}$$

The smallest (normalized) positive number

Let $s=0$, $c=1$ and $f=0$ which is equivalent to

$$2^{-1022} \cdot (1+0) \approx 2.225 \times 10^{-308}.$$

The largest number

Let $s=0$, $c=2046$ and $f=1-2^{-52}$ which is equivalent to

$$2^{1023} \cdot (2-2^{-52}) \approx 1.798 \times 10^{308}.$$

Format of floating-point number

$$(-1)^s \times (1+f) \times 2^{c-1023}$$

The smallest (normalized) positive number

Let $s=0$, $c=1$ and $f=0$ which is equivalent to

$$2^{-1022} \cdot (1+0) \approx 2.225 \times 10^{-308}.$$

The largest number

Let $s=0$, $c=2046$ and $f=1-2^{-52}$ which is equivalent to

$$2^{1023} \cdot (2-2^{-52}) \approx 1.798 \times 10^{308}.$$

Chopping and rounding

For any real number x, let

$$x = \pm 1.a_1 a_2 \cdots a_t a_{t+1} a_{t+2} \cdots \times 2^m,$$

denote the normalized scientific binary representation of x.

1. **chopping:** simply discard the excess bits $a_{t+1}, a_{t+2}, \ldots$ to obtain

$$fl(x) = \pm 1.a_1 a_2 \cdots a_t \times 2^m.$$

2. **rounding:** add $\pm 2^{-(t+1)} \times 2^m$ to x and then chop the excess bits to obtain a number of the form

$$fl(x) = \pm 1.\delta_1 \delta_2 \cdots \delta_t \times 2^m.$$

In this method, if $a_{t+1} = 1$, we add 1 to a_t to obtain $fl(x)$, and if $a_{t+1} = 0$, we merely chop off all but the first t digits.

Chopping and rounding

For any real number x, let

$$x = \pm 1.a_1a_2\cdots a_ta_{t+1}a_{t+2}\cdots \times 2^m,$$

denote the normalized scientific binary representation of x.

1. **chopping**: simply discard the excess bits $a_{t+1}, a_{t+2}, \ldots$ to obtain

$$fl(x) = \pm 1.a_1a_2\cdots a_t \times 2^m.$$

2. **rounding**: add $\pm 2^{-(t+1)} \times 2^m$ to x and then chop the excess bits to obtain a number of the form

$$fl(x) = \pm 1.\delta_1\delta_2\cdots\delta_t \times 2^m.$$

In this method, if $a_{t+1} = 1$, we add 1 to a_t to obtain $fl(x)$, and if $a_{t+1} = 0$, we merely chop off all but the first t digits.

Chopping and rounding

For any real number x, let

$$x = \pm 1.a_1 a_2 \cdots a_t a_{t+1} a_{t+2} \cdots \times 2^m,$$

denote the normalized scientific binary representation of x.

1. **chopping**: simply discard the excess bits $a_{t+1}, a_{t+2}, \ldots$ to obtain

$$fl(x) = \pm 1.a_1 a_2 \cdots a_t \times 2^m.$$

2. **rounding**: add $\pm 2^{-(t+1)} \times 2^m$ to x and then chop the excess bits to obtain a number of the form

$$fl(x) = \pm 1.\delta_1 \delta_2 \cdots \delta_t \times 2^m.$$

In this method, if $a_{t+1} = 1$, we add 1 to a_t to obtain $fl(x)$, and if $a_{t+1} = 0$, we merely chop off all but the first t digits.

Definition (Roundoff error)

The error results from replacing a number with its floating-point form is called *roundoff error* or *rounding error*.

Definition (Absolute Error and Relative Error)

If x is an approximation to the exact value $x^\star$, the absolute error is $|x^\star - x|$ and the relative error is $\frac{|x^\star - x|}{|x^\star|}$, provided that $x^\star \neq 0$.

Example

(a) If $x = 0.3000 \times 10^{-3}$ and $x^* = 0.3100 \times 10^{-3}$, then the absolute error is 0.1×10^{-4} and the relative error is 0.3333×10^{-1}.
(b) If $x = 0.3000 \times 10^{4}$ and $x^* = 0.3100 \times 10^{4}$, then the absolute error is 0.1×10^{3} and the relative error is 0.3333×10^{-1}.

Definition (Roundoff error)

The error results from replacing a number with its floating-point form is called *roundoff error* or *rounding error*.

Definition (Absolute Error and Relative Error)

If x is an approximation to the exact value $x^\star$, the absolute error is $|x^\star - x|$ and the relative error is $\frac{|x^\star - x|}{|x^\star|}$, provided that $x^\star \neq 0$.

Example

(a) If $x = 0.3000 \times 10^{-3}$ and $x^* = 0.3100 \times 10^{-3}$, then the absolute error is 0.1×10^{-4} and the relative error is 0.3333×10^{-1}.
(b) If $x = 0.3000 \times 10^{4}$ and $x^* = 0.3100 \times 10^{4}$, then the absolute error is 0.1×10^{3} and the relative error is 0.3333×10^{-1}.

Definition (Roundoff error)

The error results from replacing a number with its floating-point form is called *roundoff error* or *rounding error*.

Definition (Absolute Error and Relative Error)

If x is an approximation to the exact value $x^\star$, the absolute error is $|x^\star - x|$ and the relative error is $\frac{|x^\star - x|}{|x^\star|}$, provided that $x^\star \neq 0$.

Example

(a) If $x = 0.3000 \times 10^{-3}$ and $x^* = 0.3100 \times 10^{-3}$, then the absolute error is 0.1×10^{-4} and the relative error is 0.3333×10^{-1}.

(b) If $x = 0.3000 \times 10^{4}$ and $x^* = 0.3100 \times 10^{4}$, then the absolute error is 0.1×10^{3} and the relative error is 0.3333×10^{-1}.

Definition (Roundoff error)

The error results from replacing a number with its floating-point form is called *roundoff error* or *rounding error*.

Definition (Absolute Error and Relative Error)

If x is an approximation to the exact value $x^\star$, the absolute error is $|x^\star - x|$ and the relative error is $\frac{|x^\star - x|}{|x^\star|}$, provided that $x^\star \neq 0$.

Example

(a) If $x = 0.3000 \times 10^{-3}$ and $x^* = 0.3100 \times 10^{-3}$, then the absolute error is 0.1×10^{-4} and the relative error is 0.3333×10^{-1}.
(b) If $x = 0.3000 \times 10^{4}$ and $x^* = 0.3100 \times 10^{4}$, then the absolute error is 0.1×10^{3} and the relative error is 0.3333×10^{-1}.

Remark

As a measure of accuracy, the absolute error may be misleading and the relative error more meaningful.

Definition

In decimal expressions, the number x^* is said to approximate x to t significant digits if t is the largest nonnegative integer for which

$$\frac{|x - x^*|}{|x|} \leq 5 \times 10^{-t}.$$

Remark

As a measure of accuracy, the absolute error may be misleading and the relative error more meaningful.

Definition

In decimal expressions, the number x^* is said to approximate x to t significant digits if t is the largest nonnegative integer for which

$$\frac{|x - x^*|}{|x|} \leq 5 \times 10^{-t}.$$

Absolute and Relative Errors

- In binary expressions, if the floating-point representation $fl_{\text{chop}}(x)$ for the number x is obtained by t digits chopping, then the relative error is

$$\frac{|x - fl_{\text{chop}}(x)|}{|x|} = \frac{|0.00\cdots 0a_{t+1}a_{t+2}\cdots \times 2^m|}{|1.a_1a_2\cdots a_ta_{t+1}a_{t+2}\cdots \times 2^m|} = \frac{|0.a_{t+1}a_{t+2}\cdots|}{|1.a_1a_2\cdots a_ta_{t+1}a_{t+2}\cdots|} \times 2^{-t}.$$

The minimal value of the denominator is 1. The numerator is bounded above by 1. As a consequence

$$\left|\frac{x - fl_{\text{chop}}(x)}{x}\right| \leq 2^{-t}.$$

- In binary expressions, if the floating-point representation $fl_{\text{chop}}(x)$ for the number x is obtained by t digits chopping, then the relative error is

$$\frac{|x - fl_{\text{chop}}(x)|}{|x|} = \frac{|0.00\cdots 0a_{t+1}a_{t+2}\cdots \times 2^m|}{|1.a_1a_2\cdots a_ta_{t+1}a_{t+2}\cdots \times 2^m|} = \frac{|0.a_{t+1}a_{t+2}\cdots|}{|1.a_1a_2\cdots a_ta_{t+1}a_{t+2}\cdots|} \times 2^{-t}.$$

The minimal value of the denominator is 1. The numerator is bounded above by 1. As a consequence

$$\left|\frac{x - fl_{\text{chop}}(x)}{x}\right| \leq 2^{-t}.$$

- In binary expressions, if the floating-point representation $fl_{\text{chop}}(x)$ for the number x is obtained by t digits chopping, then the relative error is

$$\frac{|x - fl_{\text{chop}}(x)|}{|x|} = \frac{|0.00\cdots 0a_{t+1}a_{t+2}\cdots \times 2^m|}{|1.a_1a_2\cdots a_ta_{t+1}a_{t+2}\cdots \times 2^m|} = \frac{|0.a_{t+1}a_{t+2}\cdots|}{|1.a_1a_2\cdots a_ta_{t+1}a_{t+2}\cdots|} \times 2^{-t}.$$

The minimal value of the denominator is 1. The numerator is bounded above by 1. As a consequence

$$\left|\frac{x - fl_{\text{chop}}(x)}{x}\right| \leq 2^{-t}.$$

- In binary expressions, if the floating-point representation $fl_{\text{chop}}(x)$ for the number x is obtained by t digits chopping, then the relative error is

$$\frac{|x - fl_{\text{chop}}(x)|}{|x|} = \frac{|0.00\cdots 0a_{t+1}a_{t+2}\cdots \times 2^m|}{|1.a_1a_2\cdots a_ta_{t+1}a_{t+2}\cdots \times 2^m|} = \frac{|0.a_{t+1}a_{t+2}\cdots|}{|1.a_1a_2\cdots a_ta_{t+1}a_{t+2}\cdots|} \times 2^{-t}.$$

The minimal value of the denominator is 1. The numerator is bounded above by 1. As a consequence

$$\left|\frac{x - fl_{\text{chop}}(x)}{x}\right| \leq 2^{-t}.$$

- If t-digit rounding arithmetic is used and
  - $a_{t+1} = 0$, then $fl_{\text{round}}(x) = \pm 1.a_1 a_2 \cdots a_t \times 2^m$. A bound for the relative error is

$$\frac{|x - fl_{\text{round}}(x)|}{|x|} = \frac{|0.a_{t+1}a_{t+2}\cdots|}{|1.a_1a_2\cdots a_t a_{t+1}a_{t+2}\cdots|} \times 2^{-t} \leq 2^{-(t+1)},$$

    since the numerator is bounded above by $\frac{1}{2}$ due to $a_{t+1} = 0$.
  - $a_{t+1} = 1$, then $fl_{\text{round}}(x) = \pm(1.a_1a_2\cdots a_t + 2^{-t}) \times 2^m$. The upper bound for relative error becomes

$$\frac{|x - fl_{\text{round}}(x)|}{|x|} = \frac{|1 - 0.a_{t+1}a_{t+2}\cdots|}{|1.a_1a_2\cdots a_t a_{t+1}a_{t+2}\cdots|} \times 2^{-t} \leq 2^{-(t+1)},$$

    since the numerator is bounded by $\frac{1}{2}$ due to $a_{t+1} = 1$.

  Therefore the relative error for rounding arithmetic is

$$\left|\frac{x - fl_{\text{round}}(x)}{x}\right| \leq 2^{-(t+1)} = \frac{1}{2} \times 2^{-t}.$$

- If t-digit rounding arithmetic is used and
  - $a_{t+1} = 0$, then $fl_{\text{round}}(x) = \pm 1.a_1a_2\cdots a_t \times 2^m$. A bound for the relative error is

$$\frac{|x - fl_{\text{round}}(x)|}{|x|} = \frac{|0.a_{t+1}a_{t+2}\cdots|}{|1.a_1a_2\cdots a_ta_{t+1}a_{t+2}\cdots|}\times 2^{-t} \leq 2^{-(t+1)},$$

    since the numerator is bounded above by $\frac{1}{2}$ due to $a_{t+1} = 0$.
  - $a_{t+1} = 1$, then $fl_{\text{round}}(x) = \pm(1.a_1a_2\cdots a_t + 2^{-t}) \times 2^m$. The upper bound for relative error becomes

$$\frac{|x - fl_{\text{round}}(x)|}{|x|} = \frac{|1 - 0.a_{t+1}a_{t+2}\cdots|}{|1.a_1a_2\cdots a_ta_{t+1}a_{t+2}\cdots|}\times 2^{-t} \leq 2^{-(t+1)},$$

    since the numerator is bounded by $\frac{1}{2}$ due to $a_{t+1} = 1$.

Therefore the relative error for rounding arithmetic is

$$\left|\frac{x - fl_{\text{round}}(x)}{x}\right| \leq 2^{-(t+1)} = \frac{1}{2} \times 2^{-t}.$$

- If t-digit rounding arithmetic is used and
  - $a_{t+1} = 0$, then $fl_{\text{round}}(x) = \pm 1.a_1 a_2 \cdots a_t \times 2^m$. A bound for the relative error is

$$\frac{|x - fl_{\text{round}}(x)|}{|x|} = \frac{|0.a_{t+1}a_{t+2}\cdots|}{|1.a_1 a_2 \cdots a_t a_{t+1} a_{t+2} \cdots|} \times 2^{-t} \leq 2^{-(t+1)},$$

    since the numerator is bounded above by $\frac{1}{2}$ due to $a_{t+1} = 0$.
  - $a_{t+1} = 1$, then $fl_{\text{round}}(x) = \pm(1.a_1 a_2 \cdots a_t + 2^{-t}) \times 2^m$. The upper bound for relative error becomes

$$\frac{|x - fl_{\text{round}}(x)|}{|x|} = \frac{|1 - 0.a_{t+1}a_{t+2}\cdots|}{|1.a_1 a_2 \cdots a_t a_{t+1} a_{t+2} \cdots|} \times 2^{-t} \leq 2^{-(t+1)},$$

    since the numerator is bounded by $\frac{1}{2}$ due to $a_{t+1} = 1$.

  Therefore the relative error for rounding arithmetic is

$$\left| \frac{x - fl_{\text{round}}(x)}{x} \right| \leq 2^{-(t+1)} = \frac{1}{2} \times 2^{-t}.$$

- If t-digit rounding arithmetic is used and
  - $a_{t+1} = 0$, then $fl_{\text{round}}(x) = \pm 1.a_1 a_2 \cdots a_t \times 2^m$. A bound for the relative error is

$$\frac{|x - fl_{\text{round}}(x)|}{|x|} = \frac{|0.a_{t+1}a_{t+2}\cdots|}{|1.a_1a_2\cdots a_t a_{t+1}a_{t+2}\cdots|} \times 2^{-t} \leq 2^{-(t+1)},$$

since the numerator is bounded above by $\frac{1}{2}$ due to $a_{t+1} = 0$.
  - $a_{t+1} = 1$, then $fl_{\text{round}}(x) = \pm(1.a_1 a_2 \cdots a_t + 2^{-t}) \times 2^m$. The upper bound for relative error becomes

$$\frac{|x - fl_{\text{round}}(x)|}{|x|} = \frac{|1 - 0.a_{t+1}a_{t+2}\cdots|}{|1.a_1a_2\cdots a_t a_{t+1}a_{t+2}\cdots|} \times 2^{-t} \leq 2^{-(t+1)},$$

since the numerator is bounded by $\frac{1}{2}$ due to $a_{t+1} = 1$.

Therefore the relative error for rounding arithmetic is

$$\left|\frac{x - fl_{\text{round}}(x)}{x}\right| \leq 2^{-(t+1)} = \frac{1}{2} \times 2^{-t}.$$

- If t-digit rounding arithmetic is used and
  - $a_{t+1} = 0$, then $fl_{\text{round}}(x) = \pm 1.a_1 a_2 \cdots a_t \times 2^m$. A bound for the relative error is

$$\frac{|x - fl_{\text{round}}(x)|}{|x|} = \frac{|0.a_{t+1}a_{t+2}\cdots|}{|1.a_1a_2\cdots a_t a_{t+1} a_{t+2}\cdots|} \times 2^{-t} \leq 2^{-(t+1)},$$

    since the numerator is bounded above by $\frac{1}{2}$ due to $a_{t+1} = 0$.
  - $a_{t+1} = 1$, then $fl_{\text{round}}(x) = \pm(1.a_1 a_2 \cdots a_t + 2^{-t}) \times 2^m$. The upper bound for relative error becomes

$$\frac{|x - fl_{\text{round}}(x)|}{|x|} = \frac{|1 - 0.a_{t+1}a_{t+2}\cdots|}{|1.a_1a_2\cdots a_t a_{t+1} a_{t+2}\cdots|} \times 2^{-t} \leq 2^{-(t+1)},$$

    since the numerator is bounded by $\frac{1}{2}$ due to $a_{t+1} = 1$.

Therefore the relative error for rounding arithmetic is

$$\left| \frac{x - fl_{\text{round}}(x)}{x} \right| \leq 2^{-(t+1)} = \frac{1}{2} \times 2^{-t}.$$

Definition (Machine epsilon)

The floating-point representation, $fl(x)$, of x can be expressed as

$$fl(x) = x(1+\delta), \quad |\delta| \le \varepsilon_M, \tag{1}$$

where $\varepsilon_M \equiv 2^{-t}$ is referred to as the *machine epsilon*. It is equivalent to the distance from 1.0 to the next largest floating point number, and also equivalent to the least upper bound of relative error resulted from chopping.

Single precision IEEE standard floating-point format

The mantissa f corresponds to 23 binary digits (i.e., $t = 23$), the machine epsilon is

$$\varepsilon_M = 2^{-23} \approx 1.192 \times 10^{-7}.$$

This approximately corresponds to 6 accurate decimal digits

Definition (Machine epsilon)

The floating-point representation, $fl(x)$, of x can be expressed as

$$fl(x) = x(1+\delta), \quad |\delta| \le \varepsilon_M, \tag{1}$$

where $\varepsilon_M \equiv 2^{-t}$ is referred to as the *machine epsilon*. It is equivalent to the distance from 1.0 to the next largest floating point number, and also equivalent to the least upper bound of relative error resulted from chopping.

Single precision IEEE standard floating-point format

The mantissa f corresponds to 23 binary digits (i.e., $t = 23$), the machine epsilon is

$$\varepsilon_M = 2^{-23} \approx 1.192 \times 10^{-7}.$$

This approximately corresponds to 6 accurate decimal digits

Double precision IEEE standard floating-point format

The mantissa f corresponds to 52 binary digits (i.e., $t = 52$), the machine epsilon is

$$\varepsilon_M = 2^{-52} \approx 2.220 \times 10^{-16},$$

which provides between 15 and 16 decimal digits of accuracy. The matlab built-in function eps returns this value by default.

Summary of IEEE standard floating-point format

	single precision	double precision
ε_M	1.192×10^{-7}	2.220×10^{-16}
smallest positive number	1.175×10^{-38}	2.225×10^{-308}
largest number	3.403×10^{38}	1.798×10^{308}
decimal precision	6	15

Double precision IEEE standard floating-point format

The mantissa f corresponds to 52 binary digits (i.e., $t = 52$), the machine epsilon is

$$\varepsilon_M = 2^{-52} \approx 2.220 \times 10^{-16},$$

which provides between 15 and 16 decimal digits of accuracy. The matlab built-in function eps returns this value by default.

Summary of IEEE standard floating-point format

	single precision	double precision
ε_M	1.192×10^{-7}	2.220×10^{-16}
smallest positive number	1.175×10^{-38}	2.225×10^{-308}
largest number	3.403×10^{38}	1.798×10^{308}
decimal precision	6	15

Double precision IEEE standard floating-point format

The mantissa f corresponds to 52 binary digits (i.e., $t = 52$), the machine epsilon is

$$\varepsilon_M = 2^{-52} \approx 2.220 \times 10^{-16},$$

which provides between 15 and 16 decimal digits of accuracy. The matlab built-in function eps returns this value by default.

Summary of IEEE standard floating-point format

	single precision	double precision
ε_M	1.192×10^{-7}	2.220×10^{-16}
smallest positive number	1.175×10^{-38}	2.225×10^{-308}
largest number	3.403×10^{38}	1.798×10^{308}
decimal precision	6	15

- Let $\odot$ stand for any one of the four basic arithmetic operators $+$, $-$, $\star$, $\div$.
- Whenever two machine numbers x and y are to be combined arithmetically, the computer will produce $fl(x \odot y)$ instead of $x \odot y$.
- Under (1), the relative error of $fl(x \odot y)$ satisfies

$$fl(x \odot y) = (x \odot y)(1 + \delta), \quad \delta \leq \varepsilon_M, \tag{2}$$

  where ε_M is the machine epsilon. In fact, $\delta \leq \varepsilon_M/2$ if rounding is used (that is, if $fl(\cdot) = fl_{\text{round}}(\cdot)$).
- But if x, y are not machine numbers, then they must first rounded to floating-point format before the arithmetic operation and the resulting relative error becomes

$$fl(fl(x) \odot fl(y)) = (x(1 + \delta_1) \odot y(1 + \delta_2))(1 + \delta_3),$$

  where $\delta_i \leq \varepsilon_M, i = 1, 2, 3.$

Machine Epsilon

- Let $\odot$ stand for any one of the four basic arithmetic operators $+$, $-$, $\star$, $\div$.
- Whenever two machine numbers x and y are to be combined arithmetically, the computer will produce $fl(x \odot y)$ instead of $x \odot y$.
- Under (1), the relative error of $fl(x \odot y)$ satisfies

$$fl(x \odot y) = (x \odot y)(1 + \delta), \quad \delta \leq \varepsilon_M, \tag{2}$$

where ε_M is the machine epsilon. In fact, $\delta \leq \varepsilon_M/2$ if rounding is used (that is, if $fl(\cdot) = fl_{\text{round}}(\cdot)$).
- But if x, y are not machine numbers, then they must first rounded to floating-point format before the arithmetic operation and the resulting relative error becomes

$$fl(fl(x) \odot fl(y)) = (x(1 + \delta_1) \odot y(1 + \delta_2))(1 + \delta_3),$$

where $\delta_i \leq \varepsilon_M, i = 1, 2, 3.$

- Let $\odot$ stand for any one of the four basic arithmetic operators $+$, $-$, $\star$, $\div$.
- Whenever two machine numbers x and y are to be combined arithmetically, the computer will produce $fl(x \odot y)$ instead of $x \odot y$.
- Under (1), the relative error of $fl(x \odot y)$ satisfies

$$fl(x \odot y) = (x \odot y)(1+\delta), \quad \delta \leq \varepsilon_M, \tag{2}$$

where ε_M is the machine epsilon. In fact, $\delta \leq \varepsilon_M/2$ if rounding is used (that is, if $fl(\cdot) = fl_{\text{round}}(\cdot)$).

- But if x, y are not machine numbers, then they must first rounded to floating-point format before the arithmetic operation and the resulting relative error becomes

$$fl(fl(x) \odot fl(y)) = (x(1+\delta_1) \odot y(1+\delta_2))(1+\delta_3),$$

where $\delta_i \leq \varepsilon_M, i = 1, 2, 3.$

- Let $\odot$ stand for any one of the four basic arithmetic operators $+$, $-$, $\star$, $\div$.
- Whenever two machine numbers x and y are to be combined arithmetically, the computer will produce $fl(x \odot y)$ instead of $x \odot y$.
- Under (1), the relative error of $fl(x \odot y)$ satisfies

$$fl(x \odot y) = (x \odot y)(1 + \delta), \quad \delta \leq \varepsilon_M, \tag{2}$$

where ε_M is the machine epsilon. In fact, $\delta \leq \varepsilon_M/2$ if rounding is used (that is, if $fl(\cdot) = fl_{\text{round}}(\cdot)$).

- But if x, y are not machine numbers, then they must first rounded to floating-point format before the arithmetic operation and the resulting relative error becomes

$$fl(fl(x) \odot fl(y)) = (x(1 + \delta_1) \odot y(1 + \delta_2))(1 + \delta_3),$$

where $\delta_i \leq \varepsilon_M, i = 1, 2, 3.$

Example

Let $x = 0.54617$ and $y = 0.54601$. Using rounding and four-digit arithmetic, then

- $x^* = fl(x) = 0.5462$ is accurate to four significant digits since

$$\frac{|x - x^*|}{|x|} = \frac{0.00003}{0.54617} = 5.5 \times 10^{-5} \leq 5 \times 10^{-4}.$$

- $y^* = fl(y) = 0.5460$ is accurate to five significant digits since

$$\frac{|y - y^*|}{|y|} = \frac{0.00001}{0.54601} = 1.8 \times 10^{-5} \leq 5 \times 10^{-5}.$$

Example

Let $x = 0.54617$ and $y = 0.54601$. Using rounding and four-digit arithmetic, then

- $x^* = fl(x) = 0.5462$ is accurate to four significant digits since

$$\frac{|x - x^*|}{|x|} = \frac{0.00003}{0.54617} = 5.5 \times 10^{-5} \leq 5 \times 10^{-4}.$$

- $y^* = fl(y) = 0.5460$ is accurate to five significant digits since

$$\frac{|y - y^*|}{|y|} = \frac{0.00001}{0.54601} = 1.8 \times 10^{-5} \leq 5 \times 10^{-5}.$$

Example

Let $x = 0.54617$ and $y = 0.54601$. Using rounding and four-digit arithmetic, then

- $x^* = fl(x) = 0.5462$ is accurate to four significant digits since

$$\frac{|x - x^*|}{|x|} = \frac{0.00003}{0.54617} = 5.5 \times 10^{-5} \leq 5 \times 10^{-4}.$$

- $y^* = fl(y) = 0.5460$ is accurate to five significant digits since

$$\frac{|y - y^*|}{|y|} = \frac{0.00001}{0.54601} = 1.8 \times 10^{-5} \leq 5 \times 10^{-5}.$$

- The exact value of subtraction is

$$r = x - y = 0.00016.$$

But

$$r^* \equiv x \ominus y = fl(fl(x) - fl(y)) = 0.0002.$$

Since

$$\frac{|r - r^*|}{|r|} = 0.25 \leq 5 \times 10^{-1}$$

the result has only one significant digit.

- Loss of accuracy

- The exact value of subtraction is

$$r = x - y = 0.00016.$$

But

$$r^* \equiv x \ominus y = fl(fl(x) - fl(y)) = 0.0002.$$

Since

$$\frac{|r - r^*|}{|r|} = 0.25 \leq 5 \times 10^{-1}$$

the result has only one significant digit.

- Loss of accuracy

Loss of Significance

- One of the most common error-producing calculations involves the cancellation of significant digits due to the subtraction of nearly equal numbers.
- Sometimes, loss of significance can be avoided by rewriting the mathematical formula in equivalent expressions.

Example

The quadratic formulas for computing the roots of $ax^2 + bx + c = 0$, when $a \neq 0$, are

$$x_1 = \frac{-b + \sqrt{b^2 - 4ac}}{2a} \quad \text{and} \quad x_2 = \frac{-b - \sqrt{b^2 - 4ac}}{2a}.$$

Consider the quadratic equation $x^2 + 62.10x + 1 = 0$ and discuss the numerical results.

Loss of Significance

- One of the most common error-producing calculations involves the cancellation of significant digits due to the subtraction of nearly equal numbers.
- Sometimes, loss of significance can be avoided by rewriting the mathematical formula in equivalent expressions.

Example

The quadratic formulas for computing the roots of $ax^2 + bx + c = 0$, when $a \neq 0$, are

$$x_1 = \frac{-b + \sqrt{b^2 - 4ac}}{2a} \quad \text{and} \quad x_2 = \frac{-b - \sqrt{b^2 - 4ac}}{2a}.$$

Consider the quadratic equation $x^2 + 62.10x + 1 = 0$ and discuss the numerical results.

Loss of Significance

- One of the most common error-producing calculations involves the cancellation of significant digits due to the subtraction of nearly equal numbers.
- Sometimes, loss of significance can be avoided by rewriting the mathematical formula in equivalent expressions.

Example

The quadratic formulas for computing the roots of $ax^2 + bx + c = 0$, when $a \neq 0$, are

$$x_1 = \frac{-b + \sqrt{b^2 - 4ac}}{2a} \quad \text{and} \quad x_2 = \frac{-b - \sqrt{b^2 - 4ac}}{2a}.$$

Consider the quadratic equation $x^2 + 62.10x + 1 = 0$ and discuss the numerical results.

Solution

- Using the quadratic formula and 8-digit rounding arithmetic, one can obtain

$$x_1 = -0.01610723 \qquad \text{and} \qquad x_2 = -62.08390.$$

- Now we perform the calculations with 4-digit rounding arithmetic. First we have

$$\sqrt{b^2 - 4ac} = \sqrt{62.10^2 - 4.000} = \sqrt{3856 - 4.000} = 62.06,$$

and

$$fl(x_1) = \frac{-62.10 + 62.06}{2.000} = \frac{-0.04000}{2.000} = -0.02000.$$

The relative error in computing x_1 is

$$\frac{|fl(x_1) - x_1|}{|x_1|} = \frac{|-0.02000 + 0.01610723|}{|-0.01610723|} \approx 0.2417 \le 5\times 10^{-1}.$$

Solution

- Using the quadratic formula and 8-digit rounding arithmetic, one can obtain

$$x_1 = -0.01610723 \qquad \text{and} \qquad x_2 = -62.08390.$$

- Now we perform the calculations with 4-digit rounding arithmetic. First we have

$$\sqrt{b^2 - 4ac} = \sqrt{62.10^2 - 4.000} = \sqrt{3856 - 4.000} = 62.06,$$

and

$$fl(x_1) = \frac{-62.10 + 62.06}{2.000} = \frac{-0.04000}{2.000} = -0.02000.$$

The relative error in computing x_1 is

$$\frac{|fl(x_1) - x_1|}{|x_1|} = \frac{|-0.02000 + 0.01610723|}{|-0.01610723|} \approx 0.2417 \leq 5\times 10^{-1}.$$

Solution

- Using the quadratic formula and 8-digit rounding arithmetic, one can obtain

$$x_1 = -0.01610723 \qquad \text{and} \qquad x_2 = -62.08390.$$

- Now we perform the calculations with 4-digit rounding arithmetic. First we have

$$\sqrt{b^2 - 4ac} = \sqrt{62.10^2 - 4.000} = \sqrt{3856 - 4.000} = 62.06,$$

and

$$fl(x_1) = \frac{-62.10 + 62.06}{2.000} = \frac{-0.04000}{2.000} = -0.02000.$$

The relative error in computing x_1 is

$$\frac{|fl(x_1) - x_1|}{|x_1|} = \frac{|-0.02000 + 0.01610723|}{|-0.01610723|} \approx 0.2417 \leq 5 \times 10^{-1}.$$

- In calculating x_2,

$$fl(x_2) = \frac{-62.10 - 62.06}{2.000} = \frac{-124.2}{2.000} = -62.10,$$

and the relative error in computing x_2 is

$$\frac{|fl(x_2) - x_2|}{|x_2|} = \frac{|-62.10 + 62.08390|}{|-62.08390|} \approx 0.259 \times 10^{-3} \leq 5 \times 10^{-4}.$$

- In this equation, $b^2 = 62.10^2$ is much larger than $4ac = 4$. Hence b and $\sqrt{b^2 - 4ac}$ become two nearly equal numbers. The calculation of x_1 involves the subtraction of two nearly equal numbers.
- To obtain a more accurate 4-digit rounding approximation for x_1, we change the formulation by rationalizing the numerator, that is,

$$x_1 = \frac{-2c}{b + \sqrt{b^2 - 4ac}}.$$

- In calculating x_2,

$$fl(x_2) = \frac{-62.10 - 62.06}{2.000} = \frac{-124.2}{2.000} = -62.10,$$

and the relative error in computing x_2 is

$$\frac{|fl(x_2) - x_2|}{|x_2|} = \frac{|-62.10 + 62.08390|}{|-62.08390|} \approx 0.259 \times 10^{-3} \leq 5 \times 10^{-4}.$$

- In this equation, $b^2 = 62.10^2$ is much larger than $4ac = 4$. Hence b and $\sqrt{b^2 - 4ac}$ become two nearly equal numbers. The calculation of x_1 involves the subtraction of two nearly equal numbers.
- To obtain a more accurate 4-digit rounding approximation for x_1, we change the formulation by rationalizing the numerator, that is,

$$x_1 = \frac{-2c}{b + \sqrt{b^2 - 4ac}}.$$

- In calculating x_2,

$$fl(x_2) = \frac{-62.10 - 62.06}{2.000} = \frac{-124.2}{2.000} = -62.10,$$

and the relative error in computing x_2 is

$$\frac{|fl(x_2) - x_2|}{|x_2|} = \frac{|-62.10 + 62.08390|}{|-62.08390|} \approx 0.259 \times 10^{-3} \leq 5 \times 10^{-4}.$$

- In this equation, $b^2 = 62.10^2$ is much larger than $4ac = 4$. Hence b and $\sqrt{b^2 - 4ac}$ become two nearly equal numbers. The calculation of x_1 involves the subtraction of two nearly equal numbers.
- To obtain a more accurate 4-digit rounding approximation for x_1, we change the formulation by rationalizing the numerator, that is,

$$x_1 = \frac{-2c}{b + \sqrt{b^2 - 4ac}}.$$

- In calculating x_2,

$$fl(x_2) = \frac{-62.10 - 62.06}{2.000} = \frac{-124.2}{2.000} = -62.10,$$

and the relative error in computing x_2 is

$$\frac{|fl(x_2) - x_2|}{|x_2|} = \frac{|-62.10 + 62.08390|}{|-62.08390|} \approx 0.259\times 10^{-3} \leq 5\times 10^{-4}.$$

- In this equation, $b^2 = 62.10^2$ is much larger than $4ac = 4$. Hence b and $\sqrt{b^2 - 4ac}$ become two nearly equal numbers. The calculation of x_1 involves the subtraction of two nearly equal numbers.
- To obtain a more accurate 4-digit rounding approximation for x_1, we change the formulation by rationalizing the numerator, that is,

$$x_1 = \frac{-2c}{b + \sqrt{b^2 - 4ac}}.$$

Then

$$fl(x_1) = \frac{-2.000}{62.10 + 62.06} = \frac{-2.000}{124.2} = -0.01610.$$

The relative error in computing x_1 is now reduced to 6.2×10^{-4}

Example

Let

$$\begin{aligned} p(x) &= x^3 - 3x^2 + 3x - 1, \\ q(x) &= ((x-3)x+3)x - 1. \qquad \text{(nested expression)} \end{aligned}$$

Compare the function values at $x = 2.19$ with using three-digit arithmetic.

Then

$$fl(x_1) = \frac{-2.000}{62.10 + 62.06} = \frac{-2.000}{124.2} = -0.01610.$$

The relative error in computing x_1 is now reduced to 6.2×10^{-4} ■

Example

Let

$$\begin{aligned} p(x) &= x^3 - 3x^2 + 3x - 1, \\ q(x) &= ((x - 3)x + 3)x - 1. \qquad \text{(nested expression)} \end{aligned}$$

Compare the function values at $x = 2.19$ with using three-digit arithmetic.

Solution

Use 3-digit and rounding for $p(2.19)$ and $q(2.19)$.

$$\begin{aligned}
\hat{p}(2.19) &= ((2.19^3 - 3 \times 2.19^2) + 3 \times 2.19) - 1 \\
&= ((10.5 - 14.4) + 3 \times 2.19) - 1 \\
&= (-3.9 + 6.57) - 1 \\
&= 2.67 - 1 = 1.67
\end{aligned}$$

and

$$\begin{aligned}
\hat{q}(2.19) &= ((2.19 - 3) \times 2.19 + 3) \times 2.19 - 1 \\
&= (-0.81 \times 2.19 + 3) \times 2.19 - 1 \\
&= (-1.77 + 3) \times 2.19 - 1 \\
&= 1.23 \times 2.19 - 1 \\
&= 2.69 - 1 = 1.69.
\end{aligned}$$

With more digits, one can have

$$p(2.19) = g(2.19) = 1.685159$$

Hence the absolute errors are

$$|p(2.19) - \hat{p}(2.19)| = 0.015159$$

and

$$|q(2.19) - \hat{q}(2.19)| = 0.004841,$$

respectively. One can observe that the evaluation formula $q(x)$ is better than $p(x)$. ∎

With more digits, one can have

$$p(2.19) = g(2.19) = 1.685159$$

Hence the absolute errors are

$$|p(2.19) - \hat{p}(2.19)| = 0.015159$$

and

$$|q(2.19) - \hat{q}(2.19)| = 0.004841,$$

respectively. One can observe that the evaluation formula $q(x)$ is better than $p(x)$.

With more digits, one can have

$$p(2.19) = g(2.19) = 1.685159$$

Hence the absolute errors are

$$|p(2.19) - \hat{p}(2.19)| = 0.015159$$

and

$$|q(2.19) - \hat{q}(2.19)| = 0.004841,$$

respectively. One can observe that the evaluation formula $q(x)$ is better than $p(x)$. ■

Definition (Algorithm)

An algorithm is a procedure that describes a finite sequence of steps to be performed in a specified order.

Example

Give an algorithm to compute $\sum_{i=1}^{n} x_i$, where n and $x_1, x_2, \ldots, x_n$ are given.

Algorithm

INPUT	$n, x_1, x_2, \ldots, x_n$.
OUTPUT	$SUM = \sum_{i=1}^{n} x_i$.
Step 1.	Set $SUM = 0$. (Initialize accumulator.)
Step 2.	For $i = 1, 2, \ldots, n$ do
	Set $SUM = SUM + x_i$. (Add the next term.)
Step 3.	OUTPUT SUM;
	STOP

Definition (Algorithm)

An algorithm is a procedure that describes a finite sequence of steps to be performed in a specified order.

Example

Give an algorithm to compute $\sum_{i=1}^{n} x_i$, where n and $x_1, x_2, \ldots, x_n$ are given.

Algorithm

INPUT $n, x_1, x_2, \ldots, x_n$.

OUTPUT $SUM = \sum_{i=1}^{n} x_i$.

Step 1. Set $SUM = 0$. (Initialize accumulator.)

Step 2. For $i = 1, 2, \ldots, n$ do

Set $SUM = SUM + x_i$. (Add the next term.)

Step 3. OUTPUT SUM;

STOP

Definition (Algorithm)

An algorithm is a procedure that describes a finite sequence of steps to be performed in a specified order.

Example

Give an algorithm to compute $\sum_{i=1}^{n} x_i$, where n and $x_1, x_2, \ldots, x_n$ are given.

Algorithm

INPUT $n, x_1, x_2, \ldots, x_n$.
OUTPUT $SUM = \sum_{i=1}^{n} x_i$.
Step 1. Set $SUM = 0$. (Initialize accumulator.)
Step 2. For $i = 1, 2, \ldots, n$ do
    Set $SUM = SUM + x_i$. (Add the next term.)
Step 3. OUTPUT SUM;
    STOP

Definition (Stable)

An algorithm is called stable if small changes in the initial data of the algorithm produce correspondingly small changes in the final results.

Definition (Unstable)

An algorithm is unstable if small errors made at one stage of the algorithm are magnified and propagated in subsequent stages and seriously degrade the accuracy of the overall calculation.

Remark

Whether an algorithm is stable or unstable should be decided on the basis of relative error.

Definition (Stable)

An algorithm is called stable if small changes in the initial data of the algorithm produce correspondingly small changes in the final results.

Definition (Unstable)

An algorithm is unstable if small errors made at one stage of the algorithm are magnified and propagated in subsequent stages and seriously degrade the accuracy of the overall calculation.

Remark

Whether an algorithm is stable or unstable should be decided on the basis of relative error.

Definition (Stable)

An algorithm is called stable if small changes in the initial data of the algorithm produce correspondingly small changes in the final results.

Definition (Unstable)

An algorithm is unstable if small errors made at one stage of the algorithm are magnified and propagated in subsequent stages and seriously degrade the accuracy of the overall calculation.

Remark

Whether an algorithm is stable or unstable should be decided on the basis of relative error.

Example

Consider the following recurrence algorithm

$$\begin{cases} x_1 = 1, \qquad x_2 = \frac{1}{3} \\ x_{n+1} = \frac{13}{3}x_n - \frac{4}{3}x_{n-1} \end{cases}$$

for computing the sequence of $\{x_n = (\frac{1}{3})^{n-1}\}$. This algorithm is unstable.

A Matlab implementation of the recurrence algorithm is as follows:

Matlab program

```
n = 30;
x = zeros(n,1);
x(1) = 1;
x(2) = 1/3;
for ii = 3:n
    x(ii) = 13 / 3 * x(ii-1) - 4 / 3 * x(ii-2);
    xstar = (1/3)^(ii-1);
    RelErr = abs(xstar-x(ii)) / xstar;
    fprintf('x(%2.0f) = %20.8d, x_ast(%2.0f) = %20.8d,', ...
        'RelErr(%2.0f) = %14.4d \n', ii,x(ii),ii,xstar,ii,RelErr);
end
```

Result of the Matlab implementation:

n	x_n	x_n^*	RelErr
9	4.57247371e-04	4.57247371e-04	4.4359e-10
11	5.08052602e-05	5.08052634e-05	6.3878e-08
13	5.64497734e-06	5.64502927e-06	9.1984e-06
15	6.26394672e-07	6.27225474e-07	1.3246e-03
16	2.05751947e-07	2.09075158e-07	1.5895e-02
17	5.63988754e-08	6.96917194e-08	1.9074e-01
18	-2.99408028e-08	2.32305731e-08	2.2889e+00

The relative error is increased by a factor of 12 after each iteration (compare the result from $n = 9$ to $n = 11$ and from $n = 16$ to $n = 17$, etc). This is a typical example of exponential instability, where the error grows exponentially in n.

Result of the Matlab implementation:

n	x_n	x_n^*	RelErr
9	4.57247371e-04	4.57247371e-04	4.4359e-10
11	5.08052602e-05	5.08052634e-05	6.3878e-08
13	5.64497734e-06	5.64502927e-06	9.1984e-06
15	6.26394672e-07	6.27225474e-07	1.3246e-03
16	2.05751947e-07	2.09075158e-07	1.5895e-02
17	5.63988754e-08	6.96917194e-08	1.9074e-01
18	-2.99408028e-08	2.32305731e-08	2.2889e+00

The relative error is increased by a factor of 12 after each iteration (compare the result from $n = 9$ to $n = 11$ and from $n = 16$ to $n = 17$, etc). This is a typical example of exponential instability, where the error grows exponentially in n.

Question

: What is the source of this instability and where does the factor 12 come from?

Example

The general solution of the three term recursion formula $x_{n+1} = ax_n + bx_{n-1}$ is given by

$$x_n = c_1 z_1^n + c_2 z_2^n \tag{3}$$

where z_1 and z_2 are the (distinct) roots of the characteristic equation $z^2 = az + b$. In case $z_1 = z_2$, equation (3) is replaced by $x_n = c_1 z_1^n + c_2 n z_1^n$.

Definition

Suppose $\{\beta_n\} \to 0$ and $\{x_n\} \to x^*$. If $\exists\ c > 0$ and an integer $N > 0$ such that

$$|x_n - x^*| \le c|\beta_n|, \quad \forall\ n \ge N,$$

then we say $\{x_n\}$ converges to x^* with rate of convergence $O(\beta_n)$, and write $x_n = x^* + O(\beta_n)$.

Example

Compare the convergence behavior of $\{x_n\}$ and $\{y_n\}$, where

$$x_n = \frac{n+1}{n^2}, \quad \text{and} \quad y_n = \frac{n+3}{n^3}.$$

Definition

Suppose $\{\beta_n\} \to 0$ and $\{x_n\} \to x^*$. If $\exists\, c > 0$ and an integer $N > 0$ such that

$$|x_n - x^*| \le c|\beta_n|, \quad \forall\, n \ge N,$$

then we say $\{x_n\}$ converges to x^* with rate of convergence $O(\beta_n)$, and write $x_n = x^* + O(\beta_n)$.

Example

Compare the convergence behavior of $\{x_n\}$ and $\{y_n\}$, where

$$x_n = \frac{n+1}{n^2}, \quad \text{and} \quad y_n = \frac{n+3}{n^3}.$$

Solution:

Note that both

$$\lim_{n\to\infty} x_n = 0 \quad \text{and} \quad \lim_{n\to\infty} y_n = 0.$$

Let $\alpha_n = \frac{1}{n}$ and $\beta_n = \frac{1}{n^2}$. Then

$$\begin{aligned} |x_n - 0| &= \frac{n+1}{n^2} \leq \frac{n+n}{n^2} = \frac{2}{n} = 2\alpha_n, \\ |y_n - 0| &= \frac{n+3}{n^3} \leq \frac{n+3n}{n^3} = \frac{4}{n^2} = 4\beta_n. \end{aligned}$$

Hence

$$x_n = 0 + O(\frac{1}{n}) \quad \text{and} \quad y_n = 0 + O(\frac{1}{n^2}).$$

This shows that $\{y_n\}$ converges to 0 much faster than $\{x_n\}$. ■