Mathematical preliminaries and error analysis

Tsung-Ming Huang

Department of Mathematics
National Taiwan Normal University, Taiwan

August 28, 2011
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
Example 1

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

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

\[
\frac{2}{3} = (0.\overline{a_1a_2a_3\ldots})_2.
\]

Multiply by 2 to obtain

\[
\frac{4}{3} = (a_1.\overline{a_2a_3\ldots})_2.
\]

Therefore, we get \( a_1 = 1 \) by taking the integer part of both sides.
Subtracting 1, we have

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

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 using floating-point hardware.
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.

<table>
<thead>
<tr>
<th>sign of mantissa</th>
<th>8 bits exponent</th>
<th>23 bits normalized mantissa</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 1</td>
<td>8 9</td>
<td>31</td>
</tr>
</tbody>
</table>

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 \( 0 \leq c \leq 255 \).
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 2

What is the decimal number of the machine number

0100000010100000000000000000000000?

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 3
What is the decimal number of the machine number

$$0100000010011111111111111111111$$?

1. The final 23 bits specify that the mantissa is

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

$$= 0.2499998807907105.$$ 

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

$$(-1)^s 2^{e-127} (1 + f) = 2^2 \cdot (1 + 0.2499998807907105)$$

$$= 4.999999523162842.$$
Example 4

What is the decimal number of the machine number

\[
01000001010000000000000000000001?
\]

1. The final 23 bits specify that the mantissa is

\[
f = 0 \cdot 2^{-1} + 1 \cdot 2^{-2} + 0 \cdot 2^{-3} + \cdots + 0 \cdot 2^{-22} + 1 \cdot 2^{-23}
\]

\[
= 0.2500001192092896.
\]

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

\[
(-1)^s 2^{c-127}(1 + f) = 2^2 \cdot (1 + 0.2500001192092896) = 5.0000000476837158.
\]
Summary

Above three examples

<table>
<thead>
<tr>
<th>Binary</th>
<th>Decimal</th>
</tr>
</thead>
<tbody>
<tr>
<td>01000000100111111111111111111111111111111111</td>
<td>4.999999523162842</td>
</tr>
<tr>
<td>0100000010100000000000000000000000000000000</td>
<td>5</td>
</tr>
<tr>
<td>0100000010100000000000000000000000000000001</td>
<td>5.0000000476837158</td>
</tr>
</tbody>
</table>

- 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 positive number
Let $s = 0$, $c = 1$ and $f = 0$ which is equivalent 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 5
If a number $x$ with $|x| < 2^{-126} \cdot (1 + 0)$, then we say that an \textit{underflow} has occurred and is generally set to zero.
If $|x| > 2^{127} \cdot (2 - 2^{-23})$, then we say that an \textit{overflow} has occurred.
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 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

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

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 $2^{-(t+1)} \times 2^m$ 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 6 (Roundoff error)
The error results from replacing a number with its floating-point form is called *roundoff error* or *rounding error*.

Definition 7 (Absolute Error and Relative Error)
If \( x \) is an approximation to the exact value \( x^* \), the absolute error is \( |x^* - x| \) and the relative error is \( \frac{|x^* - x|}{|x^*|} \), provided that \( x^* \neq 0 \).

Example 8
(a) If \( x = 0.3000 \times 10^{-3} \) and \( x^* = 0.3100 \times 10^{-3} \), then the absolute error is \( 0.1 \times 10^{-4} \) and the relative error is \( 0.3333 \times 10^{-1} \).
(b) If \( x = 0.3000 \times 10^4 \) and \( x^* = 0.3100 \times 10^4 \), then the absolute error is \( 0.1 \times 10^3 \) and the relative error is \( 0.3333 \times 10^{-1} \).
Remark 1
As a measure of accuracy, the absolute error may be misleading and the relative error more meaningful.

Definition 9
The number $x^*$ is said to approximate $x$ to $t$ significant digits if $t$ is the largest nonnegative integer for which

$$\frac{|x - x^*|}{|x|} \leq 5 \times 10^{-t}.$$
If the floating-point representation \( fl(x) \) for the number \( x \) is obtained by using \( t \) digits and chopping procedure, then the relative error is

\[
\frac{|x - fl(x)|}{|x|} = \frac{|0.00 \ldots 0a_{t+1}a_{t+2} \ldots \times 2^m|}{|1.a_1a_2 \ldots a_t a_{t+1}a_{t+2} \ldots \times 2^m|} = \frac{|0.a_{t+1}a_{t+2} \ldots |}{|1.a_1a_2 \ldots a_t a_{t+1}a_{t+2} \ldots |} \times 2^{-t}.
\]

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

\[
\left| \frac{x - fl(x)}{x} \right| \leq 2^{-t}.
\]
Absolute and Relative Errors

- If \( t \)-digit rounding arithmetic is used and
  - \( a_{t+1} = 0 \), then \( fl(x) = \pm 1.a_1 a_2 \cdots a_t \times 2^m \). A bound for the relative error is
    \[
    \frac{|x - fl(x)|}{|x|} = \frac{|0.a_{t+1} a_{t+2} \cdots|}{|1.a_1 a_2 \cdots a_t a_{t+1} a_{t+2} \cdots|} \times 2^{-t} \leq 2^{-(t+1)},
    \]
    since the numerator is bounded above by \( \frac{1}{2} \) due to \( a_{t+1} = 0 \).
  - \( a_{t+1} = 1 \), then \( fl(x) = \pm (1.a_1 a_2 \cdots a_t + 2^{-t}) \times 2^m \). The upper bound for relative error becomes
    \[
    \frac{|x - fl(x)|}{|x|} = \frac{|1 - 0.a_{t+1} a_{t+2} \cdots|}{|1.a_1 a_2 \cdots a_t a_{t+1} a_{t+2} \cdots|} \times 2^{-t} \leq 2^{-(t+1)},
    \]
    since the numerator is bounded by \( \frac{1}{2} \) due to \( a_{t+1} = 1 \).

Therefore the relative error for rounding arithmetic is
\[
\left| \frac{x - fl(x)}{x} \right| \leq 2^{-(t+1)} = \frac{1}{2} \times 2^{-t}.
\]
Definition 10 (Machine epsilon)

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

$$fl(x) = x(1 + \delta), \quad |\delta| \leq \varepsilon_M,$$

where $\varepsilon_M \equiv 2^{-t}$ is referred to as the unit roundoff error or machine epsilon.

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.

Summary of IEEE standard floating-point format

<table>
<thead>
<tr>
<th></th>
<th>single precision</th>
<th>double precision</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\varepsilon_M$</td>
<td>$1.192 \times 10^{-7}$</td>
<td>$2.220 \times 10^{-16}$</td>
</tr>
<tr>
<td>smallest positive number</td>
<td>$1.175 \times 10^{-38}$</td>
<td>$2.225 \times 10^{-308}$</td>
</tr>
<tr>
<td>largest number</td>
<td>$3.403 \times 10^{38}$</td>
<td>$1.798 \times 10^{308}$</td>
</tr>
<tr>
<td>decimal precision</td>
<td>6</td>
<td>15</td>
</tr>
</tbody>
</table>
Let \( \odot \) stand for any one of the four basic arithmetic operators +, −, *, ÷.

Whenever two \textbf{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 \( \varepsilon_M \) is the unit roundoff.

But if \( x, y \) are \textbf{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 $x = 0.54617$ 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

\[ r^* \equiv x \oplus y = fl(fl(x) - fl(y)) = 0.0002. \]

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 or the addition of one very large number and one very small number.

- Sometimes, loss of significance can be avoided by rewriting the mathematical formula.

Example 11

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

\[ x_1 = -0.01610723 \quad \text{and} \quad x_2 = -62.08390. \]

- 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

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

\[ \frac{|fl(x_1) - x_1|}{|x_1|} = \frac{|-0.02000 + 0.01610723|}{| -0.01610723|} \approx 0.2417 \leq 5 \times 10^{-1}. \]
In calculating $x_2$,

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

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

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 \( 6.2 \times 10^{-4} \).

**Example 12**

Let

\[
\begin{align*}
p(x) &= x^3 - 3x^2 + 3x - 1, \\
q(x) &= ((x - 3)x + 3)x - 1.
\end{align*}
\]

Compare the function values at \( x = 2.19 \) with using three-digit arithmetic.
Use 3-digit and rounding for $p(2.19)$ and $q(2.19)$.

\[
\hat{p}(2.19) = ((2.19^3 - 3 \times 2.19^2) + 3 \times 2.19) - 1
= ((10.5 - 14.4) + 3 \times 2.19) - 1
= (-3.9 + 6.57) - 1
= 2.67 - 1 = 1.67
\]

and

\[
\hat{q}(2.19) = ((2.19 - 3) \times 2.19 + 3) \times 2.19 - 1
= (-0.81 \times 2.19 + 3) \times 2.19 - 1
= (-1.77 + 3) \times 2.19 - 1
= 1.23 \times 2.19 - 1
= 2.69 - 1 = 1.69.
\]
With more digits, one can have

\[ p(2.19) = g(2.19) = 1.685159 \]

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

and

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

respectively. \( q(x) \) is better than \( p(x) \).
Definition 13 (Algorithm)

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

Example 14

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 15 (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 16 (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 17

Consider the following recurrence algorithm

\[
\begin{align*}
    x_0 &= 1, \\
    x_1 &= \frac{1}{3}, \\
    x_{n+1} &= \frac{13}{3} x_n - \frac{4}{3} x_{n-1}
\end{align*}
\]

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

A Matlab implementation of the recurrence algorithm gives the following result.
<table>
<thead>
<tr>
<th>$n$</th>
<th>$x_n$</th>
<th>$x^*_n$</th>
<th>RelErr</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>4.57247371e-04</td>
<td>4.57247371e-04</td>
<td>4.4359e-10</td>
</tr>
<tr>
<td>10</td>
<td>5.08052602e-05</td>
<td>5.08052634e-05</td>
<td>6.3878e-08</td>
</tr>
<tr>
<td>12</td>
<td>5.64497734e-06</td>
<td>5.64502927e-06</td>
<td>9.1984e-06</td>
</tr>
<tr>
<td>14</td>
<td>6.26394672e-07</td>
<td>6.27225474e-07</td>
<td>1.3246e-03</td>
</tr>
<tr>
<td>15</td>
<td>2.05751947e-07</td>
<td>2.09075158e-07</td>
<td>1.5895e-02</td>
</tr>
<tr>
<td>16</td>
<td>5.63988754e-08</td>
<td>6.96917194e-08</td>
<td>1.9074e-01</td>
</tr>
<tr>
<td>17</td>
<td>-2.99408028e-08</td>
<td>2.32305731e-08</td>
<td>2.2889e+00</td>
</tr>
</tbody>
</table>

The error present in $x_n$ is multiplied by $\frac{13}{3}$ in computing $x_{n+1}$. For example, the error will be propagated with a factor of $(\frac{13}{3})^{14}$ in computing $x_{15}$. Additional roundoff errors in computing $x_2, x_3, \ldots$ may also be propagated and added to that of $x_{15}$. 
Matlab program

```matlab
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);
xn = (1/3)^(ii-1);
    RelErr = abs(xn-x(ii)) / xn;
    fprintf('x(%2.0f) = %20.8d, x\_ast(%2.0f) = %20.8d, RelErr(%2.0f) = %14.4d\n', ii,x(ii),ii,xn,ii,RelErr);
end
```
Definition 18
Suppose \( \{\beta_n\} \to 0 \) and \( \{x_n\} \to x^* \). If \( \exists \ c > 0 \) and an integer \( N > 0 \) such that
\[
|x_n - x^*| \leq c|\beta_n|, \quad \forall \ n \geq N,
\]
then we say \( \{x_n\} \) converges to \( x^* \) with rate of convergence \( O(\beta_n) \), and write \( x_n = x^* + O(\beta_n) \).

Example 19
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

\[
| x_n - 0 | = \frac{n + 1}{n^2} \leq \frac{n + n}{n^2} = \frac{2}{n} = 2\alpha_n,
\]

\[
| y_n - 0 | = \frac{n + 3}{n^3} \leq \frac{n + 3n}{n^3} = \frac{4}{n^2} = 4\beta_n.
\]

Hence

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

This shows that \( \{y_n\} \) converges to 0 much faster than \( \{x_n\} \). \( \blacksquare \)