Lecture 4: Intro to Numerical Methods

E4300: Intro to Numerical Methods
Prof. Sandro Fusco
Office Hours:
   Mondays 5:30 to 7:00pm
   (1106B Mudd Bldg.)
Lecture 4: Last week we saw ...

• Taylor’s Theorem
• Evaluation of Polynomials and Horner's method
• Basic definitions of Error Analysis (absolute error, relative error, percent error, decimal precision)
• Floating Point Error (started)
Lecture 4: Today we are going to study ...

"Floating Point Error": Errors arising from approximating real numbers with finite-precision numbers

"Truncation Error": e.g. Errors arising from approximating a function with a simpler function e.g. \( \sin(x) \approx x \)
Lecture 4: Floating Point Error ...

Binary Representation and Base 2 Arithmetic:

\[ 1010_2 = 10_{10} \]
\[ 27_{10} = ?????_2 \]

Binary Arithmetic is carried out in a similar way to Decimal Arithmetic (remember \( 1_2 + 1_2 = 10_2 \))

\[
\begin{array}{c}
11 \\
1010_2 \\
+ \quad 11011_2 \\
\hline
100101_2
\end{array}
\]
Binary Representation and Base 2 Arithmetic:

\[ 11/2_{10} = 5.5_{10} = 101.1_2 \]
\[ 1/10_{10} = 0.1_{10} = 0.0001100_2 \]

\[ \begin{array}{c}
.0001100 \\
1010 / 1.00000000 \\
1010 \\
1100 \\
1010 \\
1000
\end{array} \]
Fixed Point Representation:

**Example:**
1 bit for the sign of the number (0/+, 1/-)
16 bits for the part left to binary point
15 bits for the part right to binary point

**Limitations:**
Numbers $\geq 2^{16}$ cannot be stored
No positive number $\leq 2^{-15}$ can be stored
Floating-Point Representation:

\[ m \times 2^E \]

where \( 1 \leq m < 2 \)

Example:

\[ \frac{10}{10} = 1010_2 = 1.010_2 \times 2^3 \]

\[
\begin{array}{c|c|c}
0 & E=3 & 1.01000000000...0 \\
\end{array}
\]

\[ \frac{1}{10} = 0.0001100_2 = 1.1001100_2 \times 2^{-4} \]

NOTE: Not a floating-point number. It must be rounded.
Floating-Point Representation:

\[ m \times 2^E \]

where \( 1 \leq m < 2 \)

**Single-Precision:** Word consists of 32 bits
- 1 bit for sign, 8 bits for the exponent, 23 bits for the significand

**Double-Precision:** Word consists of 64 bits
- 1 bit for sign, 11 bits for the exponent, 52 bits for the significand

(MatLab default is double precision)
Floating-Point Representation:

$$m \times 2^E$$

where $1 \leq m < 2$

**Hidden-bit representation:**

If significand is of the form $b_0.b_1 \ldots b_{23}$, then instead of storing $b_0, b_1, \ldots, b_{22}$, we can keep an extra place by storing $b_1, \ldots, b_{23}$, knowing $b_0 = 1$.

**Example:** $10_{10} = 1010_2 = 1.010_2 \times 2^3$

<table>
<thead>
<tr>
<th>E</th>
<th>01000000000...0</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>0</td>
</tr>
</tbody>
</table>
Floating-Point Representation:

\[ m \times 2^E \]

where \( 1 \leq m < 2 \)

**Extended-Precision**: Word consists of 80 bits

1 bit for sign, 15 bits for the exponent, 64 bits for the significand.

**NOTE**: Numbers stored in extended precision do not use hidden-bit storage.
Floating-Point Representation:

\[ m \times 2^E \]

where \( 1 \leq m < 2 \)

**Machine Precision** \( \varepsilon \):
The gap between 1 and the next larger floating-point number (in MatLab: eps).

**Single-Precision**: \( \varepsilon = 2^{-23} \approx 1.2 \times 10^{-7} \)

**Double-Precision**: \( \varepsilon = 2^{-52} \approx 2.2 \times 10^{-16} \)
Example: Toy System:

\[ 1.b_1b_2 \times 2^E \]

where \( E \) can be 0, 1, and -1.

1. What are the numbers can be represented in this system?
2. Machine Precision is ?
3. Gap is larger as we move away from the origin.
IEEE Floating-Point Double Precision:

<table>
<thead>
<tr>
<th>Exponent Field</th>
<th>Number:</th>
<th>Type of Number:</th>
</tr>
</thead>
<tbody>
<tr>
<td>000000000000</td>
<td>\pm(0.b_1 \ldots b_{52})_2 \times 2^{-1022}</td>
<td>0 or subnormal</td>
</tr>
<tr>
<td>000000000001 = 1_{10}</td>
<td>\pm(1.b_1 \ldots b_{52})_2 \times 2^{-1022}</td>
<td>Normalized number</td>
</tr>
<tr>
<td>000000000010 = 2_{10}</td>
<td>\pm(1.b_1 \ldots b_{52})_2 \times 2^{-1021}</td>
<td></td>
</tr>
<tr>
<td>\vdots</td>
<td>\pm(1.b_1 \ldots b_{52})_2 \times 2^0</td>
<td></td>
</tr>
<tr>
<td>01111111111 = 1023_{10}</td>
<td>\pm(1.b_1 \ldots b_{52})_2 \times 2^{1023}</td>
<td></td>
</tr>
<tr>
<td>\vdots</td>
<td>\pm(1.b_1 \ldots b_{52})_2 \times 2^0</td>
<td></td>
</tr>
<tr>
<td>11111111110 = 2046_{10}</td>
<td>\pm \text{inf if } b_1 = \ldots = b_{52} = 0, NaN otherwise</td>
<td>Exception</td>
</tr>
<tr>
<td>11111111111</td>
<td>NaN otherwise</td>
<td></td>
</tr>
</tbody>
</table>

Special representation is needed for 0, ± inf, NaN.
Subnormal numbers have less precision, as significand is shifted right.
Smallest subnormal number is \(2^{-52} \times 2^{-1022} = 2^{-1074}\)
IEEE Rounding Standards:

Round Down: largest floating point \(\leq x\)

Round Up: smallest floating point \(\geq x\)

Round Towards 0: either round-down(x) or round-up(x), whichever lies between 0 and x.

Round to Nearest: either round-down(x) or round-up(x), whichever is closer.

Home Reading: Sections 5.5 to 5.7
"Truncation Error": e.g. Errors arising from approximating a function with a simpler function e.g. $\sin(x) \approx x$
Taylor’s Theorem: Let $f(x)$ be a function $\in C^{n+1}[a, b]$ and $x_0 \in [a, b]$. Then for every $x \in (a, b)$ there exists a number $c = c(x)$ that lies between $x_0$ and $x$ such that

$$f(x) = T_N(x) + R_N(x)$$

where

$$T_N(x) = \sum_{n=0}^{N} \frac{f^{(n)}(x_0)(x-x_0)^n}{n!}$$

and

$$R_N(x) = \frac{f^{n+1}(c)(x-x_0)^{n+1}}{(n+1)!}$$
• **Example**: Use Taylor Series expansions with \( n=0 \) to 6 to approximate \( f(x)=\cos(x) \) at \( x_{i+1}=\pi/3 \) on the basis of the value of \( f(x) \) and its derivatives at \( x_i=\pi/4 \).

**NOTE**: \( h= \pi/3 - \pi/4 = \pi/12 \)

\[
\begin{align*}
n=0 & \quad f\left(\frac{\pi}{3}\right) \approx \cos\left(\frac{\pi}{4}\right) = \frac{2}{2} = 0.707106781 \\
|\epsilon_i| &= \left| 0.5 - 0.707106781 \right| 100\% = 41.4\%
\end{align*}
\]
Lecture 4: Truncation Error...

\[ f\left(\frac{\pi}{3}\right) \equiv \cos\left(\frac{\pi}{4}\right) - \sin\left(\frac{\pi}{4}\right)\left(\frac{\pi}{12}\right) = 0.521986659 \]

\[ |\varepsilon_t| = \left| \frac{0.5 - 0.521986659}{0.5} \right| \times 100\% = 4.40\% \]

\[ n=2 \quad |\varepsilon_t| = \ldots = 0.449\% \]

etc.
Lecture 4: Truncation Error ...

• "Big O" notation:

\[ v(t_{i+1}) = v(t_i) + v'(t_i) \cdot (t_{i+1} - t_i) + R_1 \]

\[ v'(t_i) = \frac{v(t_{i+1}) - v(t_i)}{t_{i+1} - t_i} - \frac{R_1}{t_{i+1} - t_i} \]

\[ \frac{R_1}{t_{i+1} - t_i} = \frac{v''(\zeta)}{2!} \cdot (t_{i+1} - t_i) = O(t_{i+1} - t_i) \]
Rules of Error Propagation:

In "Big O" notation let

\[ f(h) = p(h) + O(h^n) \]
\[ g(h) = q(h) + O(h^m) \]

and \( r = \min(n,m) \) then:

\[ f + g = \]
\[ f * g = \]
• **Forward Finite Difference**

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

or

\[ f'(x_i) = \frac{f(x_{i+1}) - f(x_i)}{h} + O(h) \]
• **Backward Finite Difference**

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

• **Centered Finite Difference**

\[ f'(x_i) = \frac{f(x_{i+1}) - f(x_{i-1})}{2h} + O(h^2) \]
Lecture 4: Truncation Error ...

• Forward Finite Difference for Higher derivatives ...

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

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

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


• Finite Difference for Higher derivatives ...
  – Backward Version

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

– Centered Version

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