Computational Methods
CMSC/AMSC/MAPL 460

Representing numbers in floating point
Error Analysis

Ramani Duraiswami,
Dept. of Computer Science
Class Outline

• Recap of floating point representation
• Matlab demos
• Error analysis
• Forward error analysis
• Backward error analysis
Floating point representation

• Represent real numbers using a finite number of bits

• Keywords
  – Mantissa, Exponent, Sign
  – Precision
  – Machine Epsilon
  – Overflow, Underflow

• Special Numbers
  – Zero
  – Inf
  – NaN
  – Underflow
Can be written...

\[( -1 )^s \times 2^E \times 1.f \]

<table>
<thead>
<tr>
<th>sign</th>
<th>exponent</th>
<th>mantissa (significand)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>000000000000</td>
<td>000000000000......00000000000000</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$E+1023$</th>
<th>$0 &lt; E+1023 &lt; 2047$</th>
<th>$E+1023 = 2047$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$f=0$</td>
<td>0</td>
<td>$\infty$</td>
</tr>
<tr>
<td>$f\approx=0$</td>
<td>Non-normalized typically underflow</td>
<td>Floating point Numbers</td>
</tr>
</tbody>
</table>
Exponent

• IEEE-Double: stored as binary number+1023
  – Also known as biased exponent
• Occupies 11 bits
  – Decimals values range from 0 to $2^{11} - 1 = 2047$
  – Exponent values used to represent numbers range from -1022 to 1023 (1 to 2046)
  – Values -1023 and 1024 are special
• IEEE-Single: stored as binary number+127
• Occupies 8 bits
  – Decimals values range from 0 to $2^8 - 1 = 255$
  – Exponent values used to represent numbers range from -126 to 127 (1 to 254)
  – Values -127 and 128 are special
Sign and Mantissa

- Sign is 0 (positive) or 1 (negative)
- $-1^s$
- Mantissa: Has the form $1+0.f$
- Double precision
  - $f$ is a 52 bit number
  - Abs min is 0
  - max is $\frac{1}{2}+1/2^2+1/2^3+\ldots+1/2^{52}$
  - Geometric series $\frac{1/2(1-(1/2)^{52})}{(1-1/2)}$
  - $= (1-(1/2)^{52}) = 1-2.22044604925031e-016$
- single precision
  - 23 bit number in
  - Abs min 0, max is $\frac{1}{2}+1/2^2+1/2^3+\ldots+1/2^{23}$
  - Geometric series $1-(1/2)^{23}=0.99999988079071$
Precision and Machine $\varepsilon$

- Any two numbers for a given value of the exponent are separated by $(2^{-52}) \times 2^{(e-1023)} = 2^{(e-1023-52)}$
- For each value of $e$ the separation is uniform
- *machine epsilon*: $\text{eps}$ is the distance from 1 to the next larger floating-point number.
- Floatgui Matlab code
Special numbers

• Zero
  – Since numbers are written as \((-1)^s (1+f) \times 2^{(e-1023)}\) we cannot have zero
  – So zero must be specially coded
  – Choose the lowest value: \(e=0\) and \(f=0\)
  – (without this understanding the number would be \(2^{-1023} = 1.1125369292536 \times 10^{-308}\))

• Infinity
  – Corresponds to \(f=0\) and \(e=2047\)
  – Without this understanding would be \(1.79769313486232 \times 10^{309}\)

• Undefined numbers
  – If any computation tries to produce a value that is undefined even in the real number system, the result is an “exception” known as Not-a-Number, or NaN.
  – Examples include \(0/0\) and Inf-Inf.
  – NaN is represented by taking \(e = 1024\) and \(f\) nonzero.
  – “Floating point exception”
Other exceptions

- **Overflow**: calculation yields number larger than realmax
- **Underflow**: calculations yields number smaller than realmin

<table>
<thead>
<tr>
<th>Number</th>
<th>Binary</th>
<th>Decimal</th>
</tr>
</thead>
<tbody>
<tr>
<td>eps</td>
<td>$2^{-52}$</td>
<td>$2.2204 \times 10^{-16}$</td>
</tr>
<tr>
<td>realmin</td>
<td>$2^{-1022}$</td>
<td>$2.2251 \times 10^{-308}$</td>
</tr>
<tr>
<td>realmax</td>
<td>$(2^{-\text{eps}}) \times 2^{1023}$</td>
<td>$1.7977 \times 10^{308}$</td>
</tr>
</tbody>
</table>
Some numbers cannot be exactly represented

\[
\frac{1}{10} = \frac{1}{2^4} + \frac{1}{2^5} + \frac{0}{2^6} + \frac{0}{2^7} + \frac{1}{2^8} + \frac{1}{2^9} + \frac{0}{2^{10}} + \frac{0}{2^{11}} + \frac{1}{2^{12}} + \ldots
\]

\[
t = \left(1 + \frac{9}{16} + \frac{9}{16^2} + \frac{9}{16^3} + \ldots + \frac{9}{16^{12}} + \frac{10}{16^{13}} \right) \cdot 2^{-4}
\]

\[
t_1 < 1/10 < t_2
\]

Where

\[
t_1 = 2^{-4} \cdot \left(1 + \frac{9}{16} + \frac{9}{16^2} + \frac{9}{16^3} + \ldots + \frac{9}{16^{12}} + \frac{9}{16^{13}} \right)
\]

\[
t_2 = 2^{-4} \cdot \left(1 + \frac{9}{16} + \frac{9}{16^2} + \frac{9}{16^3} + \ldots + \frac{9}{16^{12}} + \frac{10}{16^{13}} \right)
\]

It turns out that 1/10 is closer to \( t_2 \) than to \( t_1 \), so \( t \) is equal to \( t_2 \). In other words,
Effects of floating point errors

• Singular equations will only be nearly singular

• Severe cancellation errors can occur

\[ 17x_1 + 5x_2 = 22 \]
\[ 1.7x_1 + 0.5x_2 = 2.2 \]

\[ \begin{bmatrix} 17 & 5 \\ 1.7 & 0.5 \end{bmatrix} \]
\[ \begin{bmatrix} 22 \\ 2.2 \end{bmatrix} \]
\[ x = A\backslash b \]

\[ x = \begin{bmatrix} -1.0588 \\ 8.0000 \end{bmatrix} \]
Measuring error

- **Absolute error** in \( c \) as an approximation to \( x \): 
  \[ |x - c| \]

- **Relative error** in \( c \) as an approximation to nonzero \( x \): 
  \[ |(x - c)/x| \]
Errors can be magnified

• Errors can be magnified during computation.
• Let us assume numbers are known to 0.05% accuracy
• Example: $2.003 \times 10^0$ and $2.000 \times 10^0$
  – both known to within ± .001
• Perform a subtraction. Result of subtraction:
  $0.003 \times 10^0$
• but true answer could be as small as $2.002 - 2.001 = 0.001$,
• or as large as $2.004 - 1.999 = 0.005$!
• Absolute error of 0.002
• Relative error of 200%!
• Adding or subtracting causes the bounds on absolute errors to be added
Error effect on multiplication/division

- Let $x$ and $y$ be true values
- Let $X=x(1+r)$ and $Y=y(1+s)$ be the known approximations
- Relative errors are $r$ and $s$
- What is the errors in multiplying the numbers?
- $XY=xy(1+r)(1+s)$
- Absolute error $=|xy(1-rs-r-s-1)| = (rs+r+s)xy$
- Relative error $= |(xy-XY)/xy|$
  $= |rs+r+s| \leq |r|+|s|+|rs|$
- If $r$ and $s$ are small we can ignore $|rs|$
- Multiplying/dividing causes relative error bounds to add
Error Analysis

- **Forward and Backward error analysis**
  - **Forward error analysis**
    - Assume that the problem we are solving is exactly specified
    - Produce an approximate answer using the algorithm considered
    - Goal of forward error analysis produce region guaranteed to contain true soln.
    - Report region and computed solution
Backward error analysis

- We know that our problem specification itself has error ("error in initial data")
- So while we think we are solving one problem we are actually solving another
- Given an answer, determine how close the problem actually solved is to the given problem.
- Report solution and input region
Well posed problems

- Hadamard postulated that for a problem to be “well posed”
  1. Solution must exist
  2. It must be unique
  3. Small changes to input data should cause small changes to solution

- Essentially this means the regions in the problem space and solution space must be small