Computer Arithmetic & Computational Errors

James R. Graham

3/19/2009
Types of Problem

- Problems can be ill-conditioned, badly posed, or sensitive
  - In the linear algebra problem tiny changes of the coefficients produce large changes in the solutions
    - See IDL examples
  - This is not the fault of the algorithm
Types of Problem

• Unstable
  – Evaluation of $\exp(x)$ for $x < 0$ using Taylor series is an unstable algorithm
    • See IDL example

• Numerical problems only have useful solutions when we have well-conditioned problems and stable algorithms.
Representation of Numbers

• Computers calculations involve
  – Integers or whole numbers
  – Floating point or real numbers

• Numbers represented internally as 1’s & 0’s
  – There is nothing natural about base 10
    • Base 12, 20, and 60 have been used by humans
Integers

• Computer integers are represented by a finite number of digits
  – E.g, 16 bit signed binary
    • -32768 (-2^{15}) to 32767 (2^{15}-1)
  – Numbers outside this range do not exist!
    • 9 - 12 = -3
    • 83 \times 16 = 48
    • 5 \div 6 = 0
    • 32767 + 1 = -32768
  – Most languages support a variety of storage
    • Unsigned 16-bit integers: 0-65,535 (2^{16}-1)
    • Long 32-bit signed integers -2,147,483,648 (-2^{31}) to +2,147,483,647 (-2^{31}-1)
    • Long long 64 bit unsigned 0-18,446,744,073,709,551,615 (2^{64}-1)
Floating Point

• Some early computers supported fixed point arithmetic
  – Essentially integer arithmetic with an imaginary decimal point

• Floating point numbers are represented as $a \times 10^b$
  – $a$ is the **mantissa** and $b$ is the **exponent**
  – $a$ is usually written with one digit to the right of the decimal and $b$ is usually an integer
  – Both $a$ and $b$ have a finite number of digits
    • There is a finite number of floating point numbers that can be represented.
Floating Point

- Two conditions are associated with the limited range of floats—there are only a finite number of values between 0 and $\infty$
  - **Overflow**: computation results in a number greater than the largest float $= \infty$
  - **Underflow**: computation results in a number that is indistinguishable from 0
    - No wrap-around with floating point numbers
- **Machine accuracy** $\varepsilon$ such that $1.0 + \varepsilon = 1.0$
  - $\varepsilon$ is not the smallest number that can be represented!
  - Every floating point operation has a **round off error** of about $\varepsilon$
    - If you are lucky, round off errors may cancel out, but there are circumstances when round off errors are cumulative