Chapter 0 Preliminaries

- Introduction
- Analysis vs. numerical analysis
- Computers and numerical analysis
- A Typical Example
- Implementing Bisection
- Computer Arithmetic and Errors
- Interval arithmetic
- Measuring the efficiency of numerical procedures

Introduction

- **Numerical Analysis**
  - The development and study of efficient and robust procedures for problems with a computer.
  - Efficiency
    - Computing & Memory
  - Robustness
- **Operations in numerical Analysis**
  - The only operations required are +, -, *, /, and comparisons
- **Characteristics of numerical solutions**
  - Always numerical
  - An approximation

Analytic vs. numerical solution

- Analytical solution vs. numerical solution
  - Example: Solving f(x)=0
  - Even an analytic solution is subject to the errors except exact arithmetic is used
- Numerical solution
  - Is always an approximation
    - Require error analysis
    - Need to consider robustness
  - Requires computers to evaluate the numerical solutions
    - Require time analysis

Computers and numerical analysis

- **Numerical solution**
  - **Algorithms or procedures**
- **Tools for the numerical solution**
  - **Programs written in Fortran, C, C++, Java,…**
  - **Computer algebra systems (good for small problems)**
    - ***Mathematica,* Maple, MATLAB**
  - **Numerical packages (good for real world problems)**
    - **IMSL, LAPACK (linear algebra system),**
    - **LINPACK, EISPPACK**

Computers and numerical analysis

- **Computer Algebra System**
  - Able to perform mathematics symbolically, also carry out numerical procedures with extreme precision
  - Good for small problems only
  - Offer an excellent learning environment
  - Very easy to use

Many numerical solutions are iterative

Example: Bisection

- Solve a root of f(x)=0 by starting with two values that enclose the root
  - At least on root in the interval if f(x) is continuous
  - Halve the interval and test for sign change

Errors in numerical procedures

- Error in original data
  - Due to measurement
  - Hard to overcome such errors, but may need to find how sensitive the results are to change in the input information (sensitive analysis)
- Human error
- Truncation error
  - Due to the method itself. Ex., truncated Taylor series of a function
- Round-off error (computers with limited precision)
- Propagated error

Errors in numerical procedures

- **Round-off error**
  - Finite representation in computers
    - Floating-point of fixed word length
  - Numbers are round when stored as floating-point numbers
- **Propagated error**
  - Errors propagated in the succeeding steps of a process due to the occurrence of an earlier error
  - It is of critical importance
    - Unstable: errors are magnified continuously, eventually overshadow the true values
    - Stable: earlier errors die out as the method continues
    - Well-conditioned vs. ill-conditioned

Absolute vs. relative error

- **Absolute error**

$$|\,\text{true value - approximate value}\,|$$

  - Not good
  - 1036.52 +- 0.010
    - accurate to 5 significant digits
    - adequate precision
  - 0.005 +- 0.010
    - bad precision
- **Relative error**

$$\frac{\text{absolute error}}{|\,\text{true value}\,|}$$

  - More independent of the scale of the value

Significant digits

- **Another term for expressing accuracy**
- **Significant digits is**
  - **How many digits in the number have meaning**
  - **A formal definition**

Let the true value have digits $d_1 d_2 d_3 \ldots . d_n d_{n+1} \ldots . d_p$

Let the approximate value have $d_1 d_2 d_3 \ldots . d_n e_{n+1} \ldots . e_p$

where $d_1 \neq 0$ and $d_{n+1} \neq e_{n+1}$.

Both values agree to n significant digits if

$$\left| d_{n+1} - e_{n+1} \right| < 5$$

Otherwise, they agree to n-1 significant digits.

Floating-point arithmetic

- Computer stores real numbers as floating-point numbers
  - 13.524 as .13524E2
  - -0.0442 as -.442E-1
- IEEE standard
  - A computer number has 3 parts
    - Sign
    - Fraction part (Mantissa)
    - Exponent part
  - Stored as a binary quantity

Floating-point arithmetic

- 3 levels of precision
  - Single
    - Length: 32
      Sign: 1,
      Mantissa: 23
      Exponent: 8,
      Range: 10^{+-38}
  - Double
    - Length: 64
      Sign: 1, Mantissa: 52
      Exponent: 11,
      Range: 10^{+-308}
  - Extended
    - Length: 80
      Sign: 1, Mantissa: 64
      Exponent: 15,
      Range: 10^{+-4931}
  - Biased exponent
    - Allows negative exponent w/o sign bit by adding a bias value to actual exponent value to make all exponents range from 0 to max
    - For single precision, bias value is 127, so -127 stored as 0, 127 as 255.

Floating-point arithmetic

- **Infinite real numbers vs. finite floating-point numbers**
  - Gap between true number and stored number
  - Results in round-off error
- **A largest number and smallest number**
  - Single precision
    - Smallest: 2.93873E-39
    - Largest: 3.40282E+38
  - Overflow, underflow

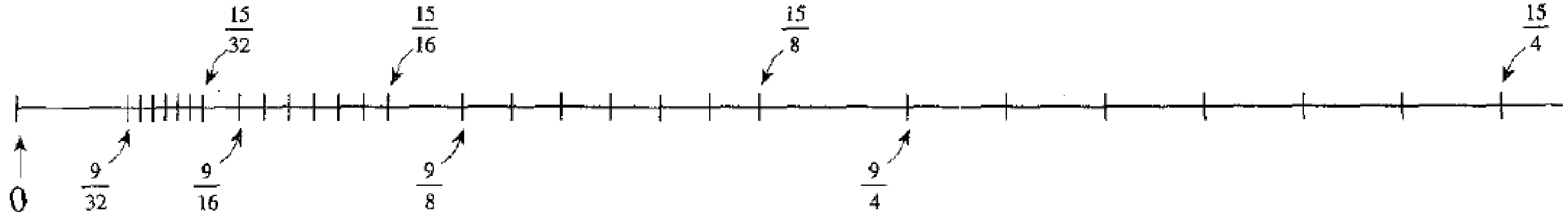
Floating-point arithmetic Examples

- **6 bits (1 for sign, 2 for exponent, 3 for mantissa)**
  - **Smallest:**
  - **Largest:**

Sign	Mantissa	Exponent	Value
0	(1)001	00	$9/16 * 2^{-1} = +9/32$
0	(1)111	11	$15/16 * 2^2 = +15/4$

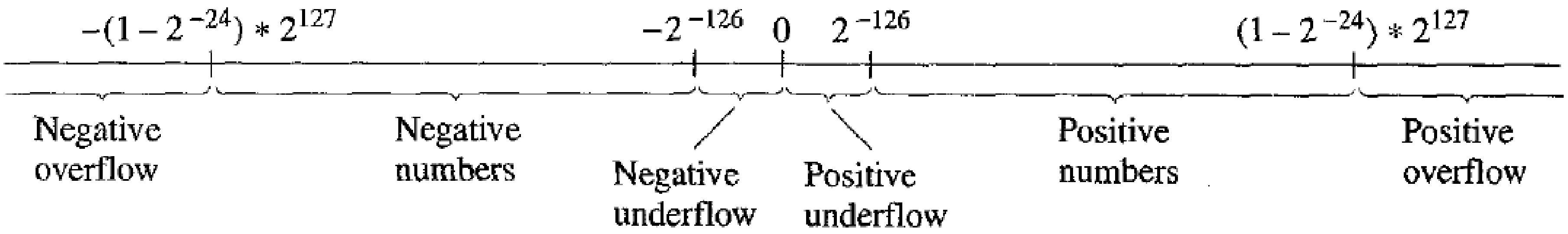
$$\text{Smallest} = 1/2 + 0/4 + 0/8 + 1/16 = 9/16,$$

$$\text{Largest} = 1/2 + 1/4 + 1/8 + 1/16 = 15/16.$$



Floating-point arithmetic Examples

- **Test in program**
  - **Don't do this: If A=B, then .....**
  - **Do this instead: If |A-B|<= TOL, then .....**



Floating-point arithmetic

- **Machine epsilon**
  - Is the smallest machine number eps such that 1+eps is not stored as 1.
  - Depends on the precision of computer system
  - For a computer that stores N-bit normalized mantissas

$$eps = \begin{cases} 2^{-N+1} & \text{if chopping is used} \\ 2^{-N} & \text{if rounding is used} \end{cases}$$

  - For single precision
    - **eps=1.192E-07=2^{-23} (rounding)**

Round-off vs. truncation error

- **Round-offer error**
  - Due to imperfect precision of the computer
  - Occurs even when the procedure is exact
- **Truncation error**
  - Caused by a procedure that does not give precise results even when the arithmetic is precise
  - Ex: evaluate f(x) using truncated Taylor series
- **Computational error**
  - Sum of round-off error and truncation error

Well-posted and well-conditioned problems

- **Accuracy of a numerical solution depends on**
  - Computer accuracy for storing number
  - Condition of the problem
  - Stability of numerical solution
    - Stable: early error are damped out as the computation proceed, they do not grow without bound
- **A problem is well-posed if**
  - A solution exists and unique, and
  - has a solution that varies continuously when values of its input vary continuously

Well-posted and well-conditioned problems

- **Condition of a problem**
  - **Well-conditioned**
    - Not sensitive to input inaccuracy
    - The change (error) in the output is not greater than the change (error) in the input
  - **Ill-conditioned**
    - Sensitive to input inaccuracy
    - A small change (error) in the input causes a large change (error) in the output
  - **Condition number C**

$$C \approx \frac{|\text{relative error in output}|}{|\text{relative error in input}|}$$

Examples

- **Compute the value in single precision**

$$\frac{(X+Y)^2 - 2XY - Y^2}{X^2} = Z,$$

X	*Y*	*Z*
0.01	1000	1.00000
0.001	1000	0.9999998
0.0001	1000	0.999213
0.00001	1000	1.000444
0.000001	1000	0.68212
0.0000001	1000	−79.58079

- **The expression can be reduced to Z=x^2/x^2 = 1**

Forward and backward error analysis

- Evaluate y=f(x)

  y_{calc} is the computed value with input x

  - Forward error

    $E_{fwd}=y_{calc}-y_{exact}$

  - Backward error

    Let x_{calc} be the x value that gives y_{calc} with no computation error

    $E_{backw}=x_{calc}-x$

Interval arithmetic

- **Interval arithmetic allows us to find how parameter errors are propagated through the sequence of computer operation of a procedure**
- **Work on intervals representing a range of numbers**

  EX: 2.4 +- 0.05 → [2.35, 2.45]
- **Rules**

$$A+B=[a_L,a_R]+[b_L,b_R]=[a_L+b_L,a_R+b_R]$$
$$A-B=[a_L,a_R]-[b_L,b_R]=[a_L-b_R,a_R-b_L]$$
$$A*B=[\min(S),\ \max(S)]$$
$$\text{where}$$
$$\mathrm{S}=\{a_L*b_L,a_L*b_R,a_R*b_L,a_R*b_R]$$

$$\text{Examples :}$$
$$[0.5,0.8]+[\text{-}1.2,0.1]=[\text{-}0.7,0.9]$$
$$[0.5,0.8]-[\text{-}1.2,0.1]=[0.4,2.0]$$
$$[0.5,0.8]-[\text{-}1.2,0.1]=[\text{-}0.96,0.08]$$

Measuring efficiency/error

- How to measure efficiency?
  - Based on operation count
  - O(n): order of n, O(n^2): order of n^2
- How to measure error?
  - Based on size of a parameter, say h. For Example,
    - Solve dy/dx=f(x,y) with a value given for y at some value for x
    - Some methods add a weighted sum of estimated values for the derivative function at evenly spaced x-values that differ by h
    - For one method the Error=(M/6)*h^3, where M depends on a value for the 3rd derivative of f(x, y)
      - O(h^3)