CS 370 - Numerical Computation **Fall 2018**

Floating Point Number Systems

Lecturer: Christopher Batty *Notes By: Harsh Mistry*

Definition 1.1 ***Infinite in extent*** *: There exists x such that verb$\|x\|$ is arbitrarily large*

Definition 1.2 ***Infinite in density*** *: Any interval $a \leq x \leq b$ contains infinitely many numbers*

- Real $\mathbb{R}$ numbers are infinite in extent and infinite in density
- Floating point systems are an approximate representation of real numbers using a finite number of bits
- Floating point numbers don't behave like true real numbers due to rounding properties and truncation
- We can express a real numbers an a infinite expansion relative to some base β
- After expressing the real number in the desired base, we multiply by a power of β to shift it into a normalized form

  $$0.d_1d_2d_3d_4 \ldots \times \beta^p$$

  where
  - d_i are diguts in base β
  - normalized implies $d_1 \neq 0$
  - exponent p is an integer

1.1 Floating Point

- Density (or precision) is bounded by limiting the number of digits t
- Extent (or range) is bounded by limited the range of values for exponent p
- The four integer parameters $\{\beta, t, L, U\}$ characterize a specific floating point system F
  - L and U bound p (The Exponent)
  - β is the base value
  - t is the number of digits the system can handle

1.2 Overflow/underflow

- if the exponent is too big or too small, our system cannot represent the number.
- When this occurs, Overflow or Underflow will occur

1.3 Rounding vs. Truncation

- **Rounding-to-nearest** - Rounds to closest available number in F
- **Truncation** - Rounds to next number in F towards zero (i.e simply discard any digits after t^th

1.4 Measuring Error

- x_{exact} = true solution
- x_{approx} = approximate solution

using these values we can distinguish between absolute error and relative error

- Absolute error
$$E_{abs} = |x_{exact} - x_{approx}|$$
- Relative error
$$E_{ret} = \frac{|x_{exact} - x_{approx}|}{|x_{exact}|}$$

1.4.1 Relative Error and Significant Digits

- Relative error is often more useful as its independent of the magnitudes of the number involved
- Relative error also relates to the number of significant digits in the result.

1.4.2 Machine Epsilon

- The maximum relative error, E, for a FP system is called machine epsilon or unit round of error.
- It defined as the smallest value such that $fl(1 + E) > 1$ under the given floating point system,
- We can derive the rule
$$fl(x) = x(1 + \delta) \text{ for some } |\delta| \leq E$$

  1
- With rounding to the nearest, $E = \frac{1}{2}\beta^{1-t}$
- With Truncation, $E = \beta^{1-t}$
- We care about the magnitude of E

1.4.3 Arithmetic with Floating Point

$$w \oplus z = fl(w + z) = (w + z)(1 + \delta)$$

1.4.4 Catastrophic Cancellation Error

Catastrophic cancellation occurs when subtracting numbers of about the same magnitude, and the input numbers contain error. All significant digits cancelled out; the result might have no correct digits whatsoever! However, if input quantities are known to be exact, we have our usual guarantee on floating point ops (addition, subtraction, etc.):

$$w \ominus z = (w - z)(1 + \delta) \text{ for } w, z, \in F$$

1.5 Conditioning of Problems

- Problems may be ill-conditioned or well-conditioned
- For problem P, with input I and output O, of a change to the input, ΔI, gives a small change in the output ΔO, P is well-conditioned. Otherwise, P is ill-conditioned
- This is a property of the problem it self and independent of any any specific implementation

1.5.1 Stability of an Algorithm

If any initial error in the data is magnified by an algorithm, the algorithm is considered numerically unstable.

1.5.2 Conditioning v.s. Stability

- Conditioning of a problem is how sensitive the problem is itself to errors/changes in input
- Stability of a numerical algorithm is how sensitive the algorithm is to errors/changes in input.
- An algorithm can be unstable even when the problem is well conditioned
- An ill-conditioned problem limits how well we can expect any algorithm to perform.

1.6 Summary

- F is not $\mathbb{R}$
- We can use **round-off error analysis** to roughly bound the errors incurred by floating point operations.
- We can analyse whether initial errors grow or shrink to determine the stability of algorithms.