Numerical Methods; October–November 2023
Assignment 2
Due: Friday, 28 November 2023

Floating-point representation

▶ Write the fine structure constant, the speed of light (in cm s^{-1}), Newton's constant (in $cm^3\ g^{-1}\ s^{-2}$), Planck's constant (in eV s), Boltzmann constant (in eV K^{-1}), Hubble constant (in s^{-1}), and the mass of the sun (in g) in 32-bit and 64-bit IEEE 754.

▶ Write these 64-bit IEEE 754 numbers in decimal:

```
0 10000001010 1001001100000000000000000000000000000000000000000000
1 10000001010 1001001100000000000000000000000000000000000000000000
0 01111111111 0101001100000000000000000000000000000000000000000000
0 01111111111 0101001100000000000000000000000000000000000000000001
```

▶ What is the maximum integer one can store in 2 bytes? 4 bytes? 8 bytes?

Round-off error

▶ What is the machine precision of the `longdouble`, `float16`, `float32`, `float64`, and `float128` data types in Numpy? Does Numpy follow the IEEE 754 standard?

▶ What is the machine precision of the `float` data type in Python?

▶ What is the machine precision for the C `float` data type on your computer? What is the relative error in storing π on your computer using this data type? What is the relative error in storing 2.0 on your computer using this data type? When the `float` data type is used, is π a machine number on your computer? Is 2.0 a machine number?

▶ Compute the absolute and relative error when the following numbers are represented using a floating point on a decimal computer with 4 digits for the mantissa and 2 digits for the exponent: π, e, $8!$, $\sqrt{2}$, 10^π.

▶ While programming in C or Python, is it OK to evaluate the expression "`x == 3.0`" for flow control? Why?

▶ Spend ten minutes browsing through the documentation of the GNU Multiple Precision Arithmetic Library (`gmplib.org`). What does this library do?

▶ Go through the documentation of the `decimal` Python module.[1] When should one use this module?

[1] `https://docs.python.org/3/library/decimal.html`

▶ Spend ten minutes browsing through the web site of the `mpmath` library (`mpmath.org`).

Floating-point arithmetic

▶ For $a = 0.23371258 \times 10^{-4}$, $b = 0.33678429 \times 10^{2}$, and $c = -0.33677811 \times 10^{2}$, calculate $a + (b + c)$ and $(a + b) + c$ in floating-point arithmetic on a decimal computer with 8-digit mantissa.

▶ Assuming a digital computer with a 3-digit mantissa, evaluate $133 + 0.921$, $133 - 0.499$, $(121 - 0.327) - 119$, and $(121 - 119) = 0.327$. In each case, compute the relative and absolute error in the result.

Propagation of errors

▶ Under what conditions is the algorithm $\phi = a + b + c$ well-conditioned?

▶ Develop two algorithms to evaluate $a^2 - b^2$ for any real numbers a and b. Compute the round-off error in both algorithms. Which of the two algorithms is more trustworthy? Which of the two algorithms is stable?

▶ Consider the quadratic equation $ax^2 + bx + c = 0$. Algorithm A computes the roots of this equation as

$$\frac{-b \pm \sqrt{b^2 - 4ac}}{2a}.$$

Compute the round-off error for this algorithm.

Algorithm B computes the roots as

$$\frac{2c}{-b \pm \sqrt{b^2 - 4ac}}.$$

Compute the round-off error for this algorithm.

Algorithm C computes the roots as q/a and c/q where

$$q = -\frac{1}{2}\left[b + \mathrm{sgn}(b)\sqrt{b^2 - 4ac}\right].$$

Compute the round-off error for this algorithm.

Of A, B, and C, which algorithm is numerically most trustworthy? Which algorithm is numerically stable?

Analysis of algorithms

▶ In Assignment 1, you wrote a code to calculate the first n Fibonacci numbers. What is the complexity of your code as a function of n? Write your answer using the O notation.