2. Overflow and underflow. Explain these concepts in the context of floating point arithmetic (based on the IEEE754-standard). Present your solution to Exercise 4.11 in Moler.

Numerical Analysis E2021

Institute of Mathematics
Aalborg University

Motivation

Numerical Analysis
E2021

Motivation

Floating Point Arithmetic

Special Values

Examples

Spacing

Underflow & overflow

- Computers can only have finite representations
- Standardisation
- Studying algorithms
  - Best answers, speed, and reliable.
- Ensuring arithmetic results are accurate and reliable

Floating Point Arithmetic

binary64 is a IEEE754 double-precision binary floating-point format specified by

- Base of $b = 2$,
- Precision of $p = 53$, where 52 bits are stored explicitly,
- Exponent range of -1022 to 1023.

all 64-bit double-precision numbers are then of the form

$$(-1)^s(1.b_{51}b_{50}\dots b_0)_2 \times 2^{e-1023} \tag{1}$$

where we use 1 bit for the sign, 52 on the mantissa, and 11 on the exponent. Note that we here have an exponent bias.

Thus we only have a finite number of floating-point numbers, which are a subset of the rationals.

Special Values

- Zero
  - is signed
- Infinity
  - is signed
- NaN
  - qNaN
  - sNaN
  - Not signed
- Subnormals
  - Numbers smaller than the smallest positive number.

Examples

$$\underbrace{0}_{\text{sign}}\ \underbrace{01111111111}_{\text{exponent}}\ \underbrace{0000000000000000000000000000000000000000000000000000}_{\text{mantissa}} = +2^{0} \times 1 = 1$$

$$\underbrace{0}_{\text{sign}}\ \underbrace{01111111111}_{\text{exponent}}\ \underbrace{0000000000000000000000000000000000000000000000000001}_{\text{mantissa}} = +2^{0} \times (1 + 2^{-52})$$

$$\underbrace{0}_{\text{sign}}\ \underbrace{11111111111}_{\text{exponent}}\ \underbrace{0000000000000000000000000000000000000000000000000000}_{\text{mantissa}} = +\infty$$

$$\underbrace{1}_{\text{sign}}\ \underbrace{11111111111}_{\text{exponent}}\ \underbrace{0000000000000000000000000000000000000000000000000000}_{\text{mantissa}} = -\infty$$

Spacing

Within each binary interval $2^e \leq x \leq 2^{e+1}$ all numbers will be equally spaced with an increment of 2^{e-52}. Thus

$$1 + \varepsilon = 1 \Rightarrow \varepsilon = 2^{-52} \tag{2}$$

This also leads to the highest relative error when rounding to be $\frac{\varepsilon}{2}$.

MATLAB Demo of `floatgui`.

A simple way to introduce large relative errors is by computing the difference between two nearly equal floating-point numbers.

MATLAB Demo of exercise 1.34

Underflow & overflow

Due to the construction of our numbers we are limited in how large, or how small, in absolute values, a floating-point number can be.

- Underflow
  - Absolute value of a non-zero result is less than `realmin`.
- Overflow
  - Absolute value of a result is greater than the largest floating-point number.
  - Results in $\pm\infty$
- Division by zero.

MATLAB Demo of exercise 4.11