# Errors

1) "Experimental"
Particularly for simulations, where we average over an ensemble of "runs", we need to pay attention to statistical errors.

2) truncation error - arising from cutting off a Taylor series expansion in a numerical scheme.

3) roundoff error - only a finite set of real numbers are exactly represented on a computer because of finite precision

single precision - 32 bits

| sign | exponent | hidden | | mantissa |
|---|---|---|---|---|
| 1 bit | 8 bits | 1 | . | $b_1 b_2 \dots b_{23}$ 23 bits |

mantissa $m$: $1 \le m < 2$

$$1.000..00 \le m \le 1.111\dots1$$

$$\pm\, 1.\underbrace{b_1 b_2 \dots b_{23}}_{\text{23 bicemals (binary decimals)}} \times 2^{[1 \text{ to } 254] - 127}$$

127 ← bias

↳ (000..0) - zero, subnormals

↳ (111 1) - inf, -inf, NaN

The $24^{th}$ bicemal can't be stored!

BTW: Biggest number is $(2 - 2^{-23}) \times 2^{254-127}$

$$\doteq 2^{128} = 3.4 \times 10^{38}$$

$$2^{128} = 10^{x} \rightarrow 128 \ln 2 = x \ln 10$$

$$x = 128 \, \ln 2 / \ln 10 \simeq 38.53$$

$$2^{128} \doteq 10^{38.53} = 10^{0.53}\, 10^{38} \doteq 3.4 \times 10^{38}$$

Machine epsilon

Adding $2^{-24}$ to $1.\underbrace{000\ldots000}_{23 \text{ bicemals}}$ yields $1.\underbrace{000\ldots000}_{23}$

$2^{-24} \approx 5.96 \times 10^{-8}$

is called machine epsilon $\epsilon_M$

$\epsilon_M$ - biggest number you can add to unity with the result rounding to unity

- also called unit roundoff

A number $1.b_1 b_2 \ldots$ can not be specified more precisely than $\epsilon_M$.

For double precision (64 bits), mantissa is $1.b_1 b_2 \ldots b_{52}$, so $\epsilon_M = 2^{-53} \doteq 1.11 \times 10^{-16}$

A real number $x$ is rounded to $\bar{x}$

$$\bar{x} = x + \epsilon x$$
$$\bar{x} = x(1+\epsilon) \qquad \text{with } |\epsilon| < \epsilon_M$$

Subtraction:

$$res = x_1 - x_2$$
$$\overline{res} = \bar{x}_1 - \bar{x}_2 + \alpha(x_1 - x_2)$$
$$\text{with } |\alpha| < \epsilon_M$$
$$= x_1(1+\epsilon_1) - x_2(1+\epsilon_2) + \alpha(x_1 - x_2)$$

$$\overline{res} = x_1 - x_2 + x_1\epsilon_1 - x_2\epsilon_2 + \alpha(x_1 - x_2)$$

$\epsilon_1$ and $\epsilon_2$ can have opposite signs
and can consider $|\epsilon_1| \approx |\epsilon_2| \approx \epsilon_M$

For $x_1 \simeq x_2$ we can write

$$\overline{res} \doteq res + 2\epsilon_M x_1 + \alpha(x_1 - x_2)$$

relative error is

$$\frac{\overline{res} - res}{res} \doteq \frac{2\epsilon_M x_1}{x_1 - x_2} + \alpha$$

can ignore

- relative roundoff error is large when $x_1 \approx x_2$

i.e. precision is reduced

# Numerical Calculus - using Taylor series

## Differentiation

recall $f(x+h) = f(x) + hf'(x) + \frac{h^2}{2} f''(x) + \ldots + \frac{h^n}{n!} f^{(n)}(x) + \ldots$

solve for $f'(x) = \frac{f(x+h) - f(x)}{h} - \left[ \frac{h}{2} f''(x) + \ldots + \frac{h^{n-1}}{n!} f^{(n)}(x) + \ldots \right]$

as $h$ becomes small, largest term in [ ] is $\frac{h}{2} f''(x)$

so we can write forward difference formula

$$f'(x) = \frac{f(x+h) - f(x)}{h} + \mathcal{O}(h)$$

$\mathcal{O}(h)$ means truncation error we are making is of order $h$

$g$ is $\mathcal{O}(h)$ if $\lim_{h \to 0} g/h = \text{constant}$

Error $\sim h$ implies that if we decrease $h$ by a factor of, say, 10, error will go down by a factor of 10.

→ compact notation: $f_i = f(x)$, $f_{i+1} = f(x+h)$, $f_{i-1} = f(x-h)$

Forward diff. formula: $f_i' = \frac{f_{i+1} - f_i}{h}$