Computational Physics

Experimental physics

Computational physics

Theoretical physics

1. Discrized analytical calculations
2. Data treatment (ERN)
3. Algorithmic modeling.

→ Computational fluid dynamics

Navier - Stokes equations

↓

Discretization

Numerical calculations.

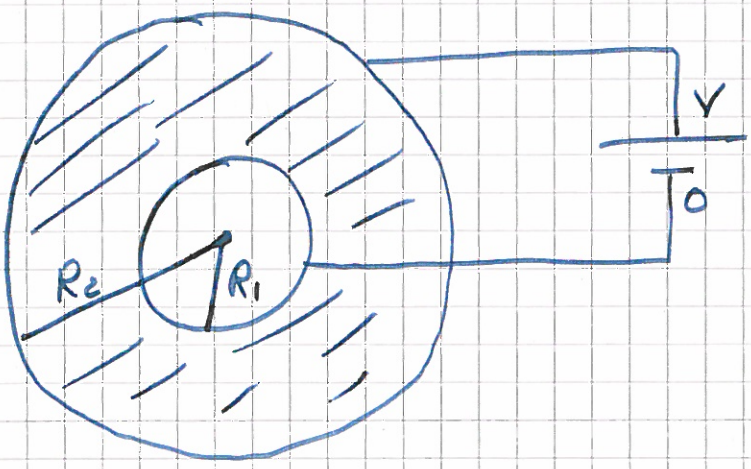
Two levels of description:

Nature → Continuous equations

→ discrete numerical models.

Typical problem:

Find electrical field inside an annulus with conducting inner and outer edges kept at fixed potentials.



Laplace equation:

$$\nabla^2 V = 0$$

Boundary conditions:

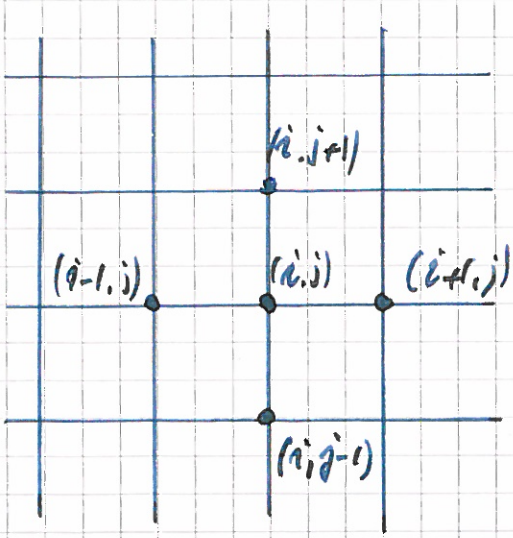
$$\left.\begin{aligned} V(\vec{R}_1) &= 0 \\ V(\vec{R}_2) &= V \end{aligned}\right\}$$

Continuous description

↓

Discretization

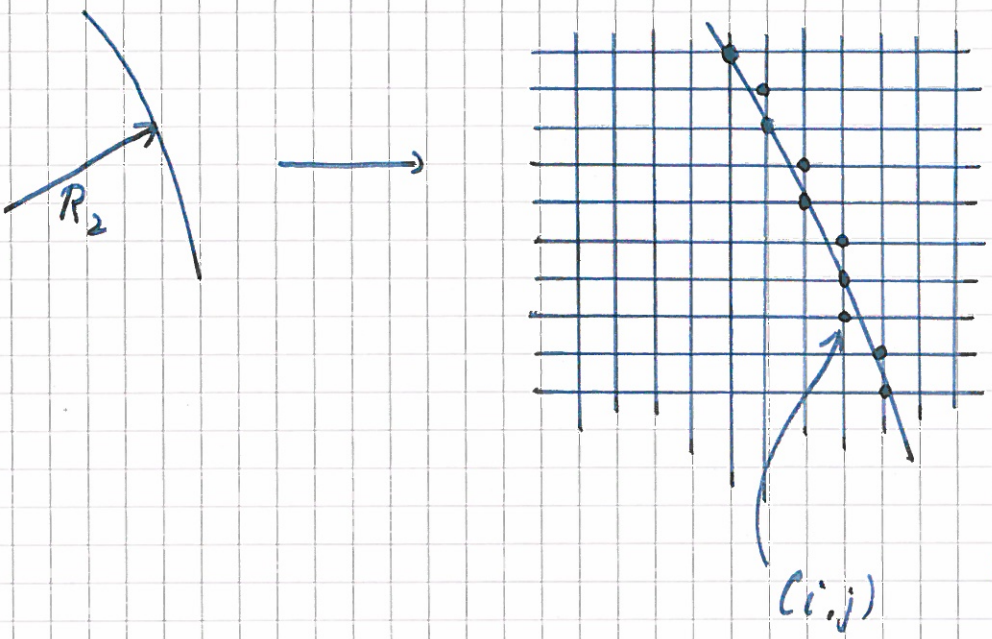
$$\nabla^2 V(\vec{r}) \rightarrow$$



$$V_{i,j+1} + V_{i,j-1} + V_{i-1,j} + V_{i+1,j} - 4V_{i,j} = 0$$

Discretized set of equations.

Boundary conditions:



This results in a <u>matrix equation:</u>

$$V_{i,j+1} + V_{i,i-1} + V_{i-1,i} + V_{i+1,i} - 4V_{ij} = 0$$

Whenever V_{ij} appearing here corresponds to a grid point on the boundary, move it to the right hand side.

$\Rightarrow$

$$\vec{A} \cdot \vec{V} = \vec{B}$$

$$\uparrow$$

$$\begin{pmatrix} V_{1,1} \\ V_{1,2} \\ \vdots \end{pmatrix}$$

Use a matrix solver $\Rightarrow$

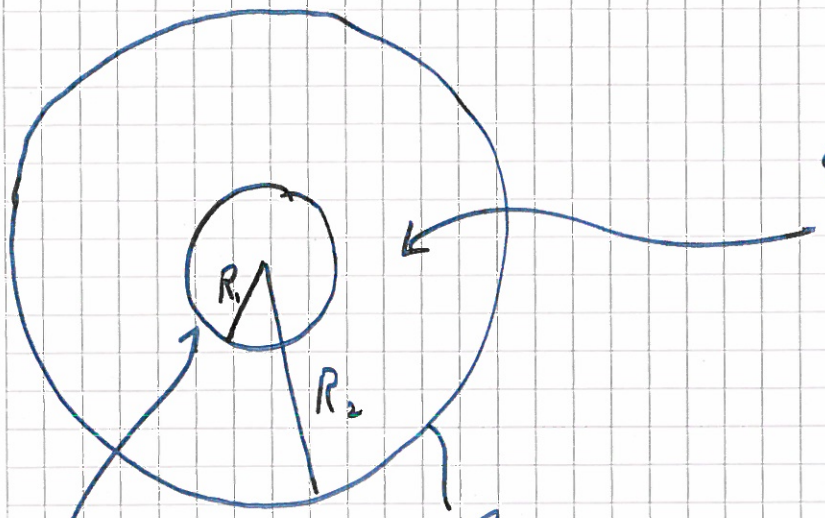
problem solved.

We now consider the following related problem:

Diffusion-limited Aggregation.

DLA

Small (colloidal) particles diffusing in a liquid near a sticky wall.



Concentration c of particles

Boundary with fixed concentration C.

Sticky boundary. Flux the concentration is zero.

(because all particles near the boundary gets absorbed by it.)

$$\left.\begin{aligned} \nabla^2 &= 0 \\ C(\vec{R}_1) &= 0 \\ C(\vec{R}_2) &= C \end{aligned}\right\}$$

BUT, what if we take into account that the sticky boundary grows when the particles stick to it.

Simplest assumption:

> Growth rate of surface is proportional to concentration gradient just outside.

If $\vec{S}$ is the surface:

$$\dot{\vec{S}} \propto \vec{\nabla} C$$

We are now dealing with a differential equation whose boundary conditions change with the solution.

It turns out that this problem is unstable on all scales:

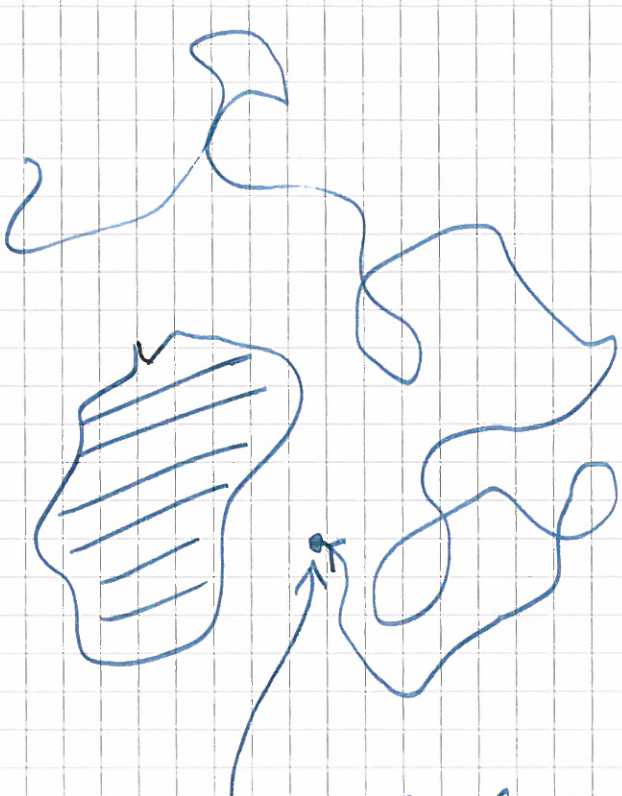
> The problem cannot be solved through solving a discretized differential equation.

This is where algorithmic modeling comes in:

Value → discrete models

↑

no analytical intermediary.

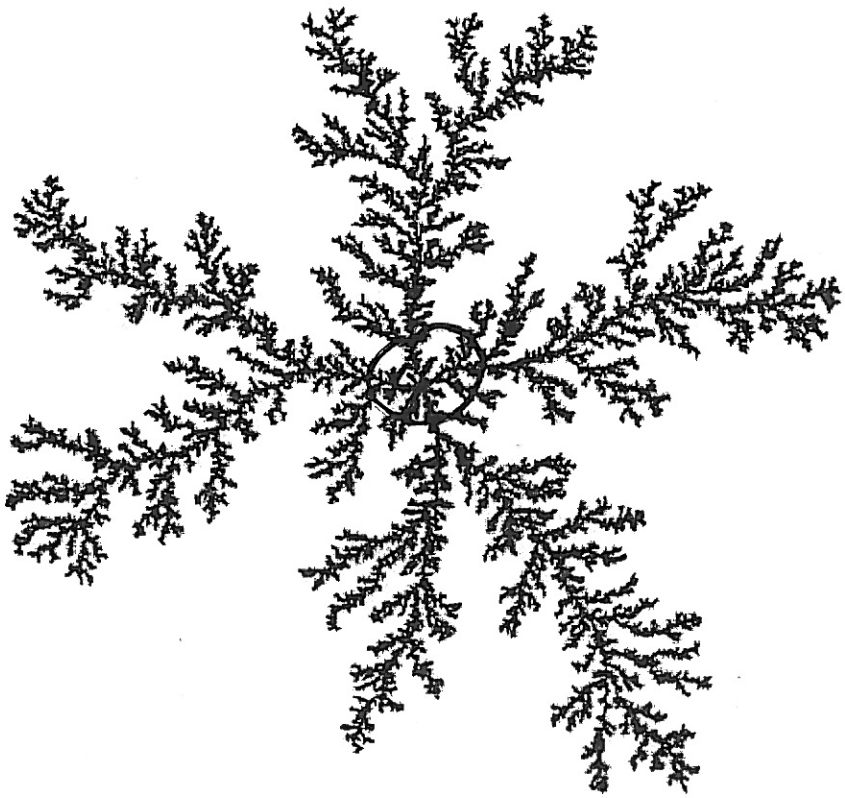


particle doing a random walk.
When it hits the surface, it sticks.

This model renders a description that fits quantitatively with what is seen in nature.

Algorithmic modeling:

Model nature directly.



$M \propto r^{D} \leftarrow 1.7$

DLA – diffusion-limited aggregation.

Another example:

Bak - Sneppen model for punctuated equilibrium.

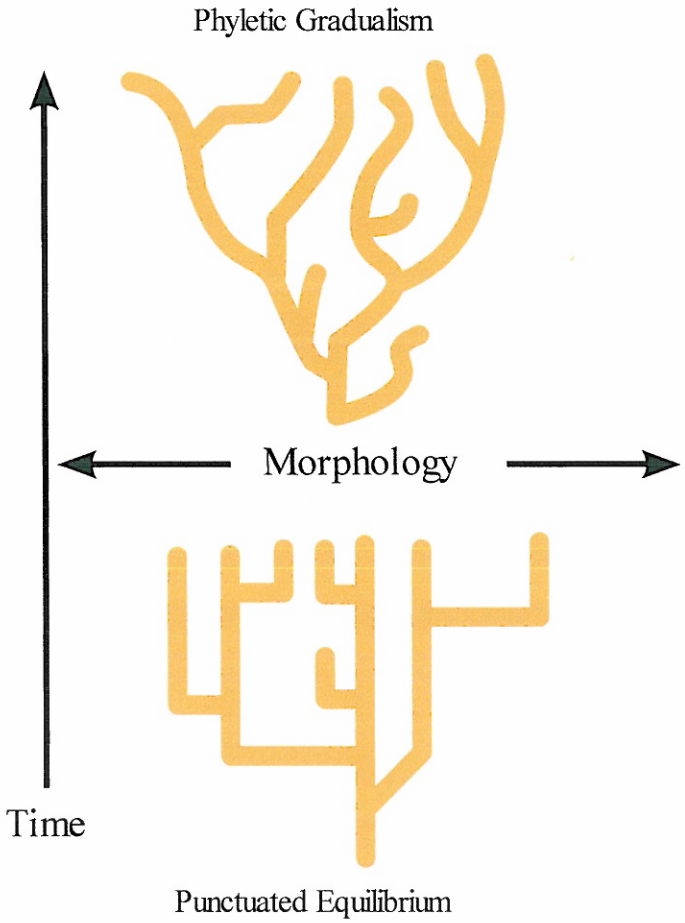
Evolution (paleontology):

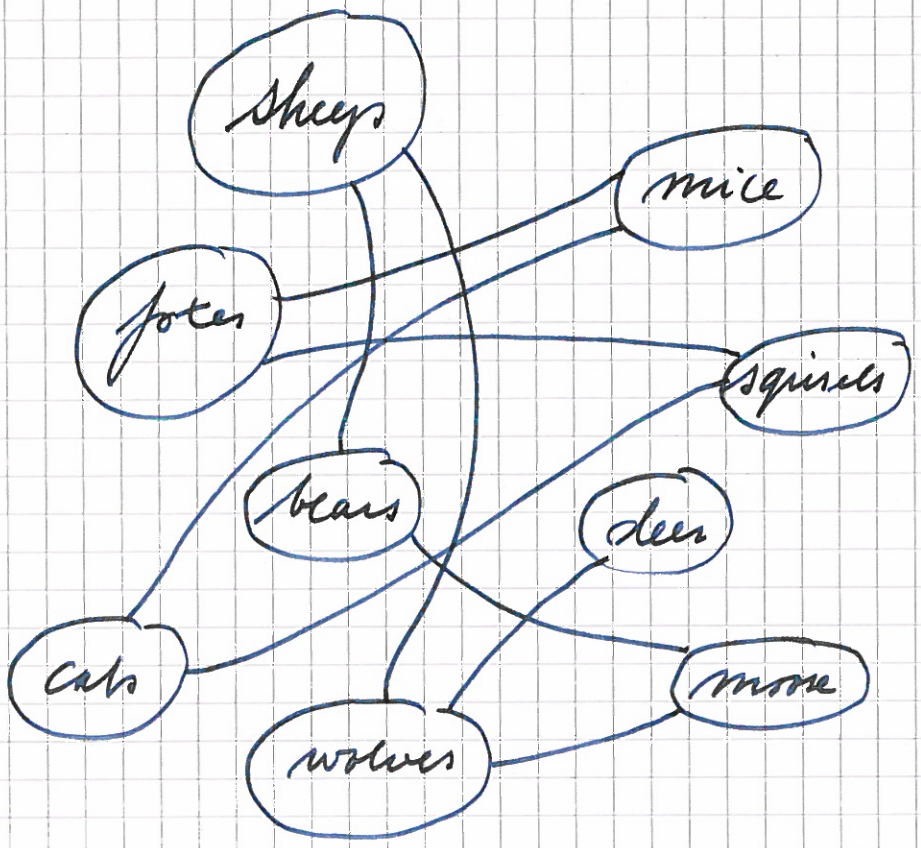
Evolution is not smooth but proceeds in a series of bursts.

Are the bursts due to internal instabilities or are external catastrophes necessary?

(e.g. extinction of dinosaurs)

Bak - Sneppen model:


Phyletic Gradualism
Morphology
Time
Punctuated Equilibrium



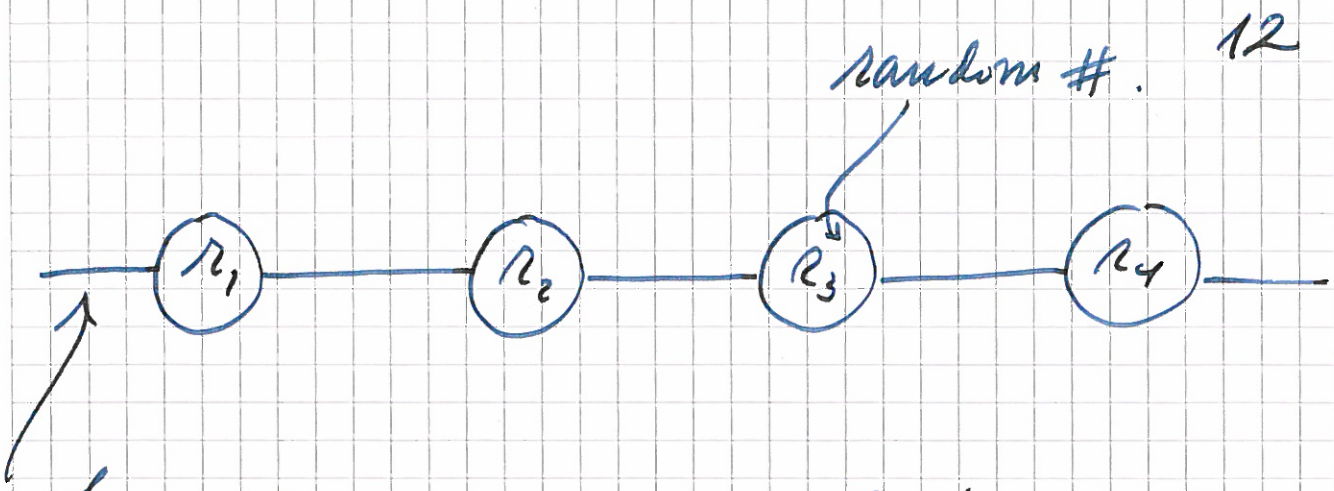
Each species is given a "fitness parameter."

The higher, the better.

Species $i \rightarrow r_i$

$\uparrow$

fitness parameter.



Simplify to a one-dimensional chain.

r_i – fitness parameter.

Scheme dynamics.

How to update the system:

(1) $r_s = \min_i r_i$, $r_{i(s)} = r_s$

(2) $r_{i(s)+1}$ → new random $r_{i(s)+1}$

$r_{i(s)}$ → new random $r_{i(s)}$

$r_{i(s)-1}$ → new random $r_{i(s)-1}$

Some elementary stuff on what computers do with numbers.

(Necessary knowledge to be able to judge what the computer produces)

How numbers are represented

Number → word

~

Consists of a string of bits.

May have length

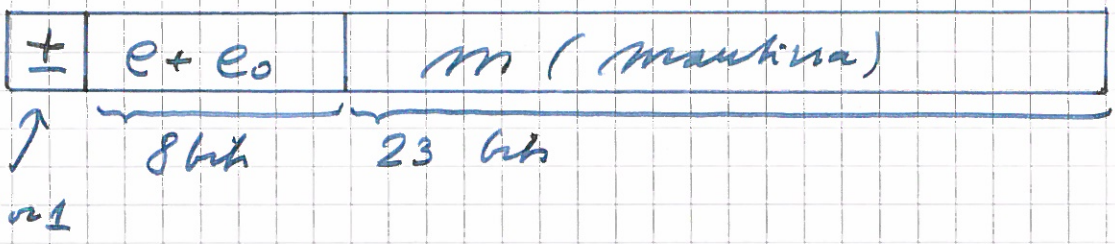
32 or 64 or ...

We will assume 32-bit architecture.

This means that only a limited range of numbers may be represented with infinite precision.

Otherwise, <u>always an approximation.</u>

$$x = \pm m \cdot 2^{e+e_0}$$



m is normalized: shoved to the left as far as possible (and e is thus given).

Since the bit to the left of m is always 1, it is typically not stored.

This gives m a precision of 24 rather than 23 bits.

e_0 is a machine-specific integer that is added to e to ensure that $e_0 + e$ is always positive.

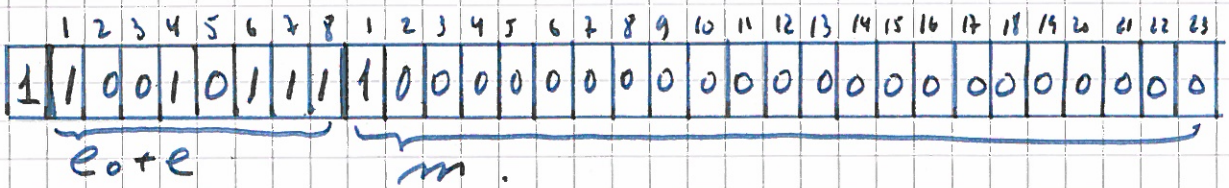
e_0 is called the <u>bias</u>.

Example:

With $e_0 = 151$, the number 0.25 is stored as:

$$0.25 = \frac{1}{4} = (1 \cdot 2^{22}) \cdot 2^{\underbrace{-24}_{e}}$$

$$-24 + 151 = 127$$



~~151~~ $= 1 \cdot 128 + 0 \cdot 64 + 0 \cdot 32 + 1 \cdot 16 + 0 \cdot 8$

$127 = 64 + 32 + 16 + 8 + 4 + 2 + 1 \quad + 1 \cdot 4 + 1 \cdot 2 + 1 \cdot 1$

$= 10010111$

↑ Binary.

How the computer performs elementary operations.

Addition

$x_1 + x_2$

$$\left.\begin{aligned} x_1 &= \pm m_1 \cdot 2^{e_1+e_0} \\ x_2 &= \pm m_2 \cdot 2^{e_2+e_0} \end{aligned}\right\} \text{Assume } e_1 \le e_2.$$

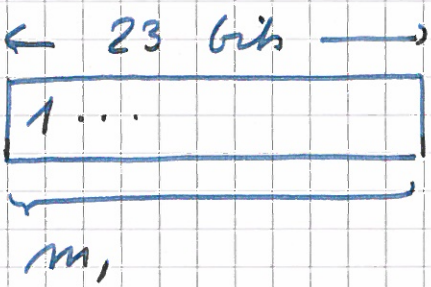
The exponent and mantissa of x_1 is adjusted so that its new exponent e_1 is equal to that of x_2:

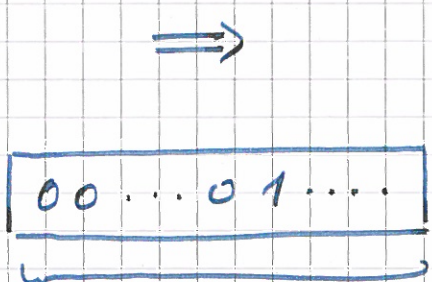
$$x_1 = \pm m_1 \cdot 2^{e_1+e_0} = m_1' \cdot 2^{e_2+e_0}$$

$$m_1' = m_1 \cdot 2^{e_1-e_2}.$$

The addition operation is then performed as:

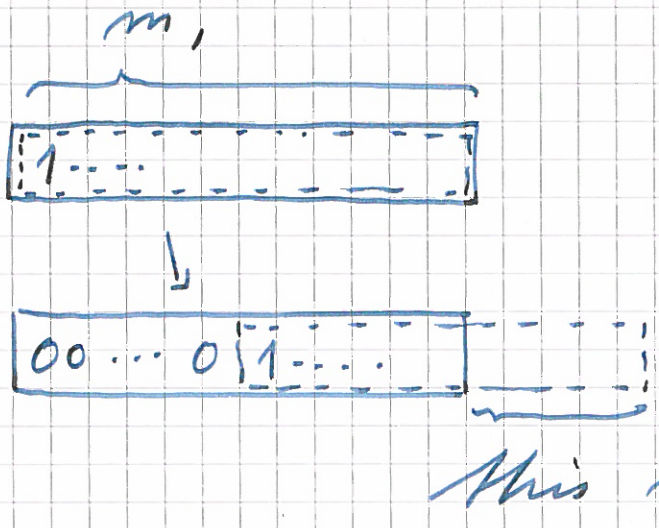
$$x_1 + x_2 = \left[(\pm m_1) + (\pm m_2) \right] e^{e_2 + e_0}$$





The adjustment has pushed the mantissa to the right.

This results in loss of the less significant bits.



This is roundoff error.

Definition of machine precision:

> Smallest number that can be added to 1 giving a result different from 1.

Example:

$2^{-22} = 2.38 \cdot 10^{-7}$

	+	129	100 ... 000	1
+	+	107	100 ... 000	2^{-22}
=				
	+	129	100 ... 000	1
+	+	129	00 ... 001	2^{-22}
=	+	129	100 ... 01	

But $1 + 2^{-23} = 1$

Subtraction works using the same principles.

However, the subtraction of two almost equal numbers is a dangerous operation.

(Since division corresponds to subtraction of logarithms, the same is true for this case.

Example: