C'era una volta …
il computing analogico

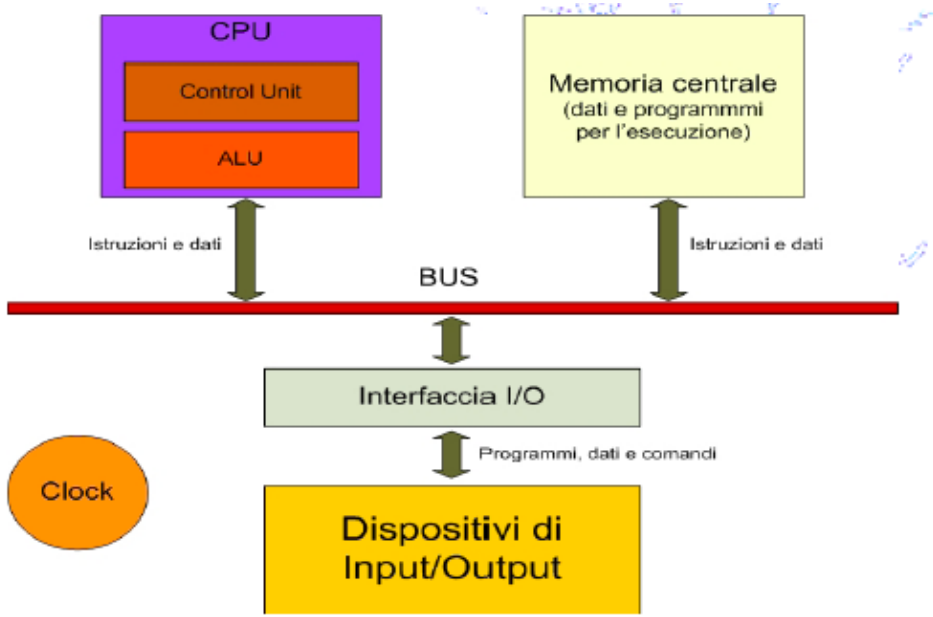
Gaetano Salina
INFN, Sezione di Roma II

Algorithm.

(webster.com)

A procedure for solving a mathematical problem (as of finding the greatest common divisor) in a finite number of steps that frequently involves repetition of an operation.

Broadly: a step-by-step procedure for solving a problem or accomplishing some end especially by a computer.

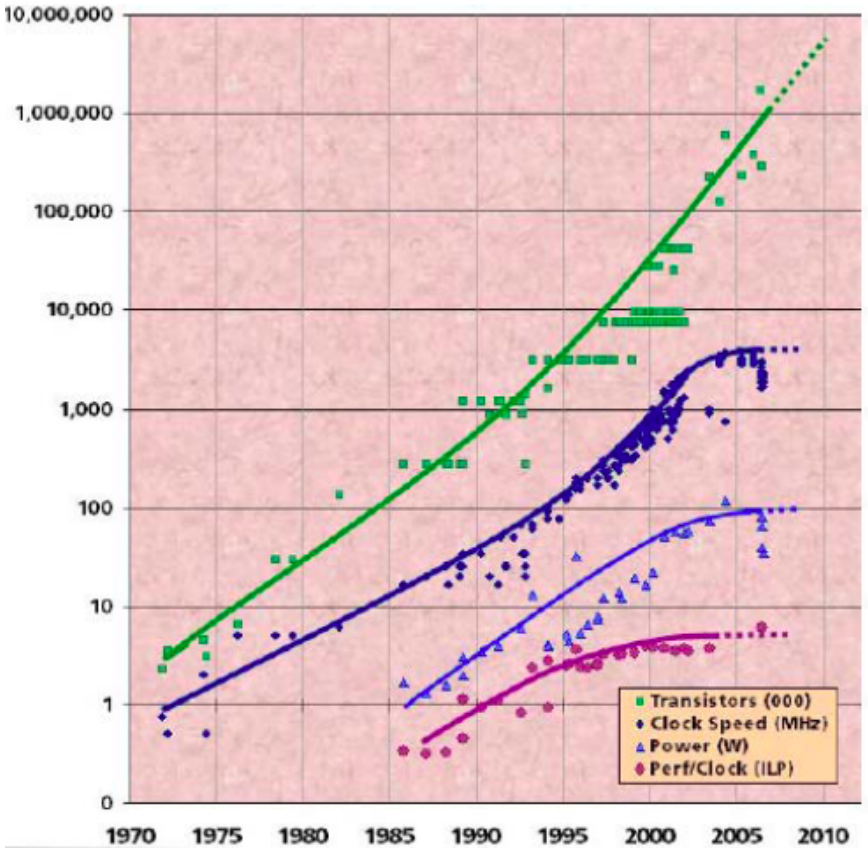


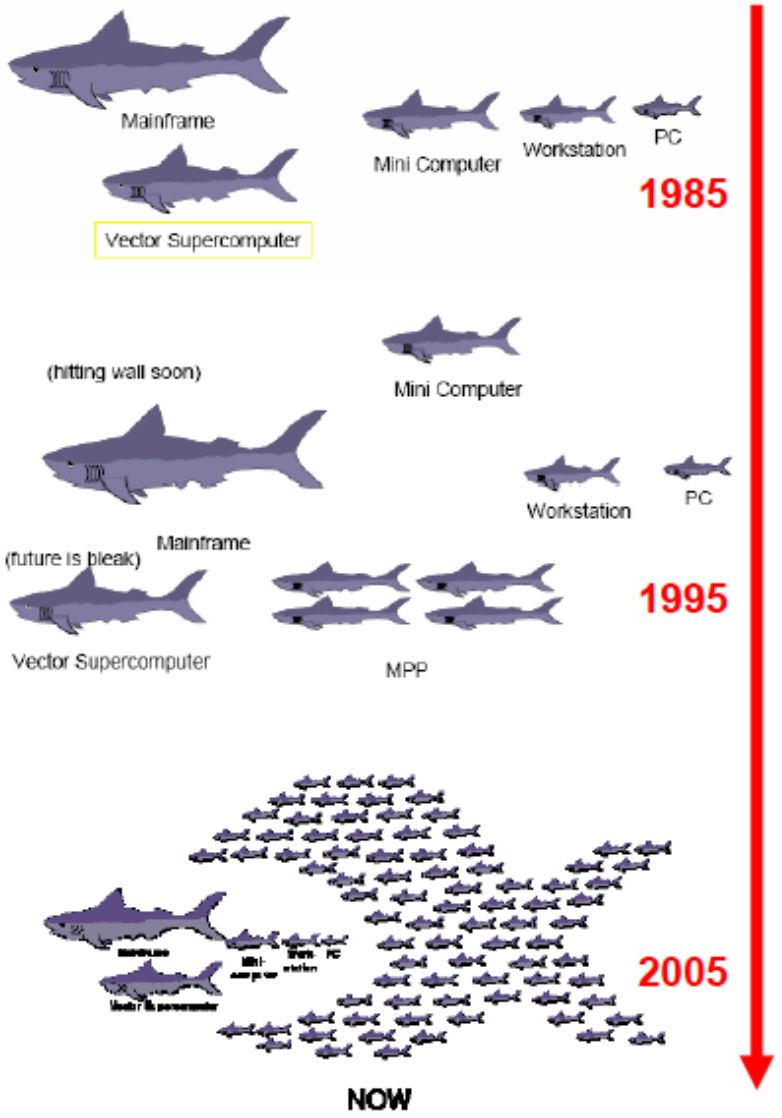
$$\left\{ \vec{x}_i(t_k) \right\} \rightarrow \left\{ \vec{F}_i(t_k) \right\}$$

$$k = k + 1 \qquad t_{k+1} = t_k + \Delta t$$

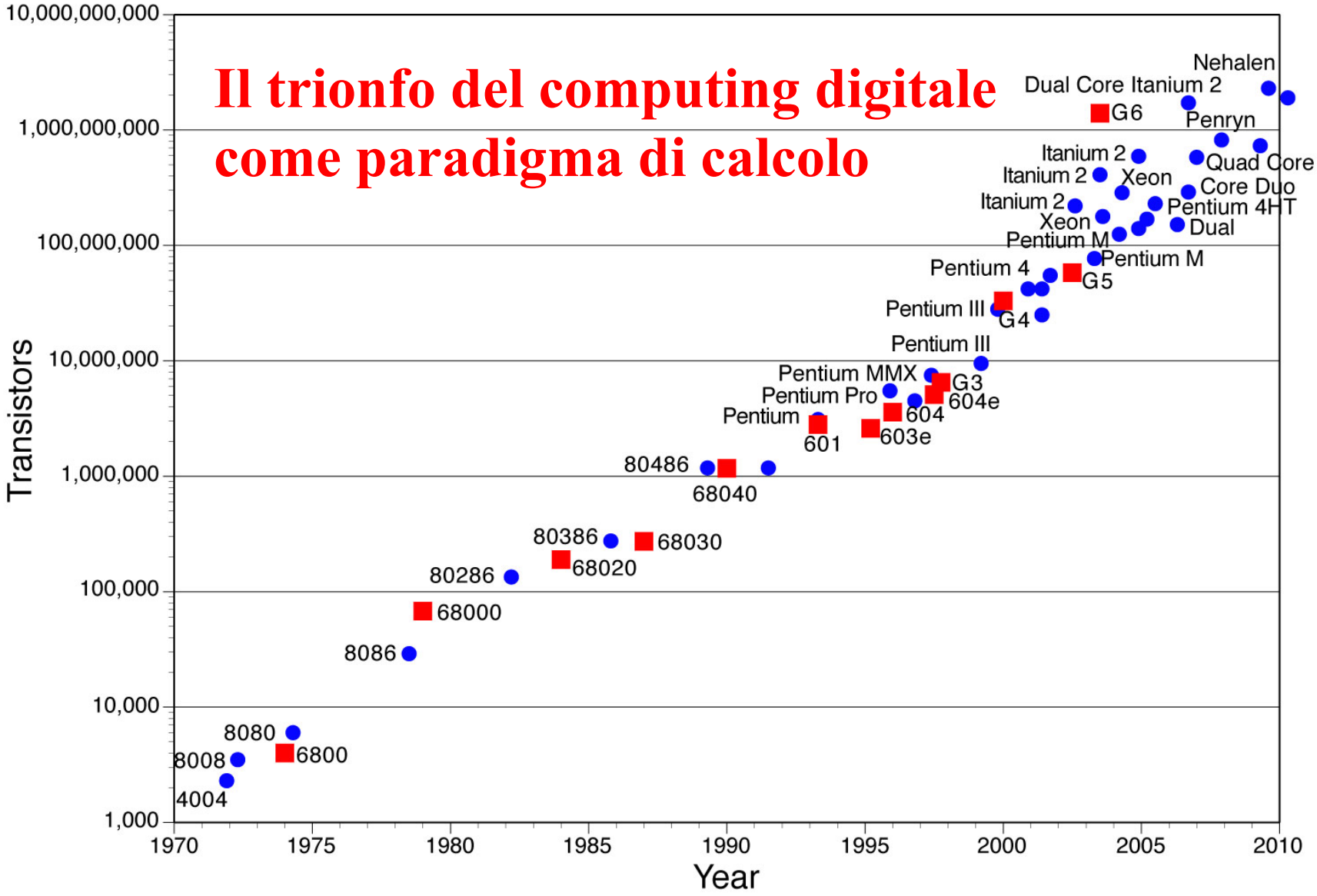
$$\left\{ \vec{x}_i(t_{k+1}) \right\} \qquad \left\{ \vec{v}_i(t_{k+1}) \right\}$$

Il trionfo del computing digitale come paradigma di calcolo





Il trionfo del computing digitale come paradigma di calcolo



Chi e' un personaggio della Walt Disney Inc ?

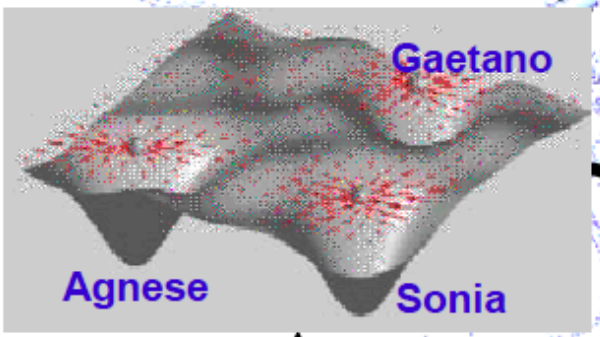


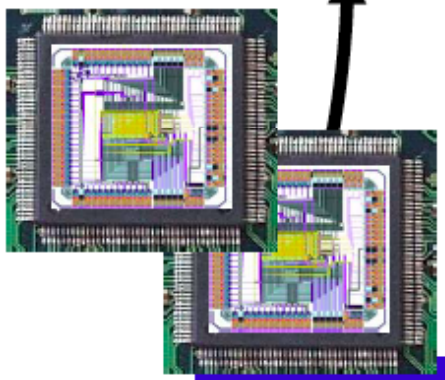





Neuroni Impulsati & Modelli Cognitivi

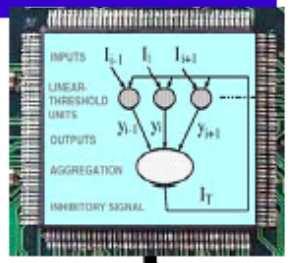
Processi di classificazione in tempo reale biologicamente plausibili.

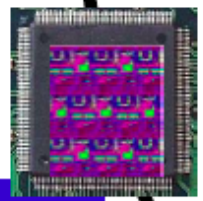




a-VLSI Networks

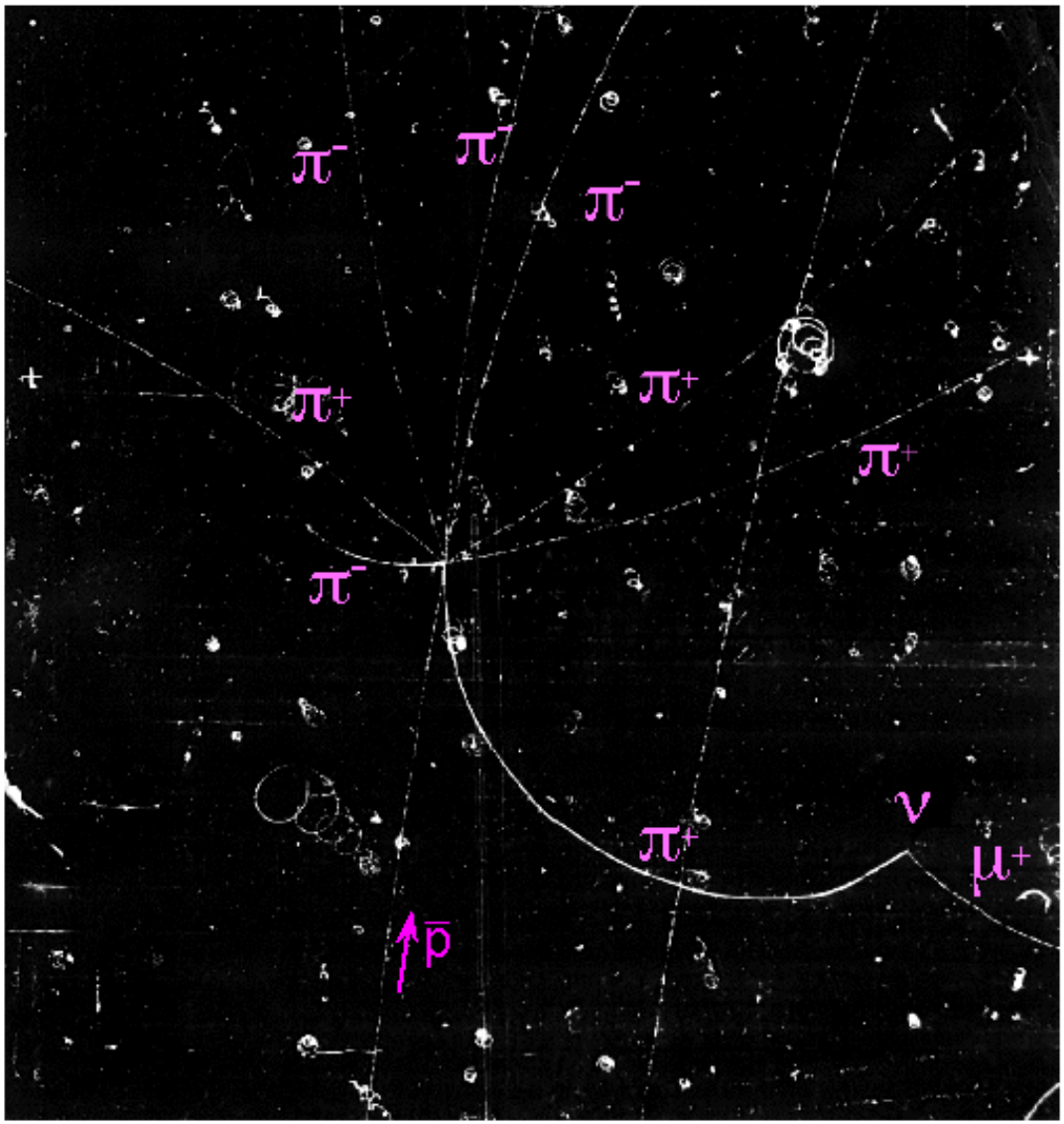
WTA Network





Retina

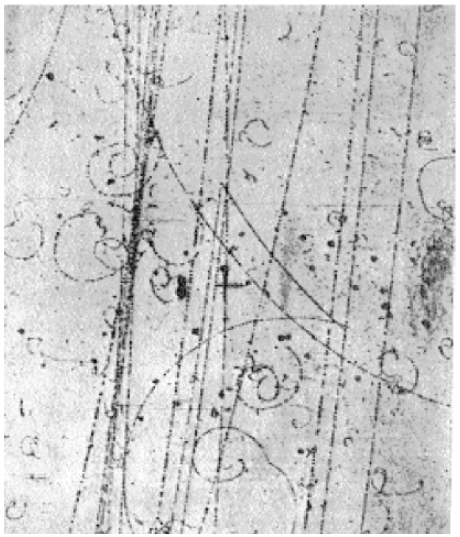
Agnese!



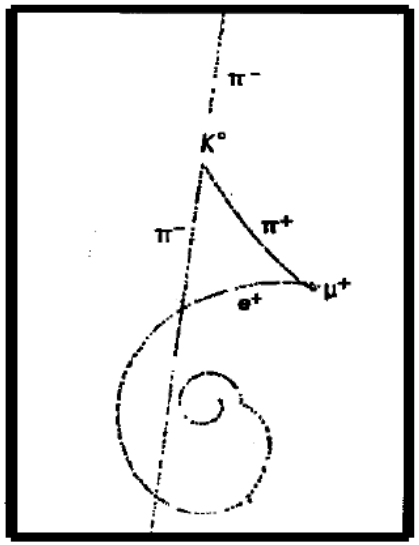
Camera a bolle:
un liquido mantenuto vicino al suo punto di ebollizione. Si riduce di colpo la pressione in modo che la temperatura del liquido sia superiore a quella di ebollizione a quella data pressione. Ma l'istante dell'abbassamento di pressione è tale che l'ebollizione inizia solo a causa degli ioni originati da particelle che passano in quel momento.

- bubble chamber
  - Vessel, filled (e.g.) with liquid hydrogen at a temperature above the normal boiling point but held under a pressure of about 10 atmospheres by a large piston to prevent boiling.
  - When particles have passed, and possibly interacted in the chamber, the piston is moved to reduce the pressure, allowing bubbles to develop along particle tracks.
  - After about 3 milliseconds have elapsed for bubbles to grow, tracks are photographed using flash photography. Several cameras provide stereo views of the tracks.
  - The piston is then moved back to recompress the liquid and collapse the bubbles before boiling can occur.
- Invented by Glaser in 1952 (when he was drinking beer)

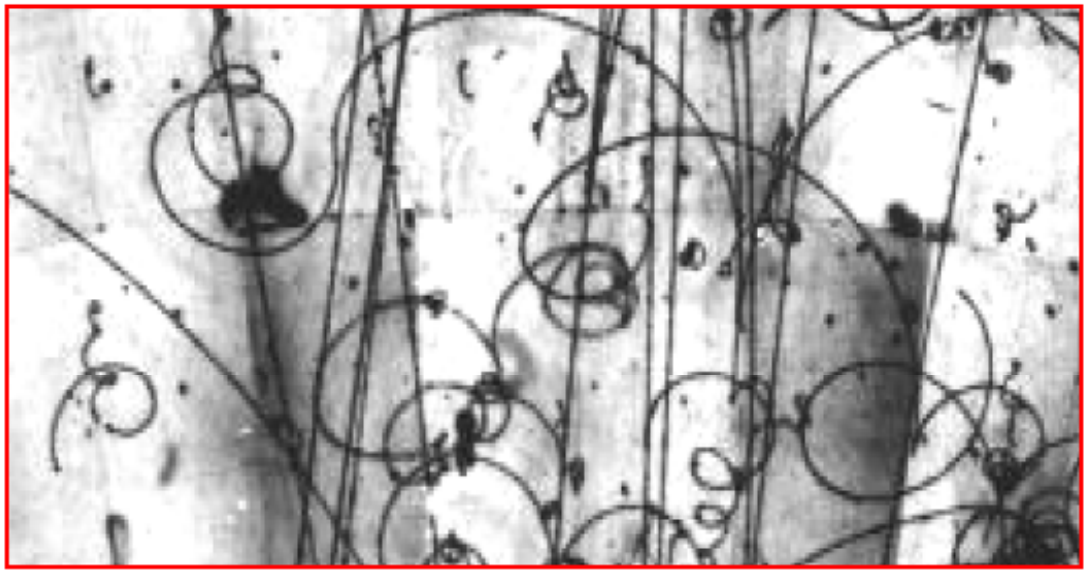
- Kaon: discovered 1947; first called "V" part



K^0 production and decay in a bubble chamber



What is the relationship between the movement of the deflected particles with their charge and the direction of the applied magnetic field.

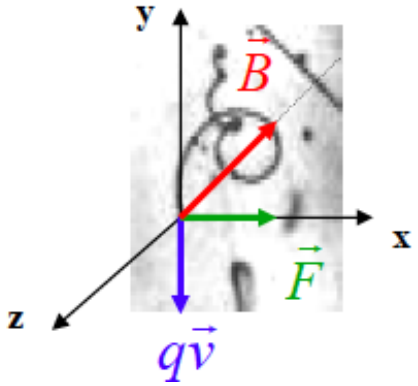


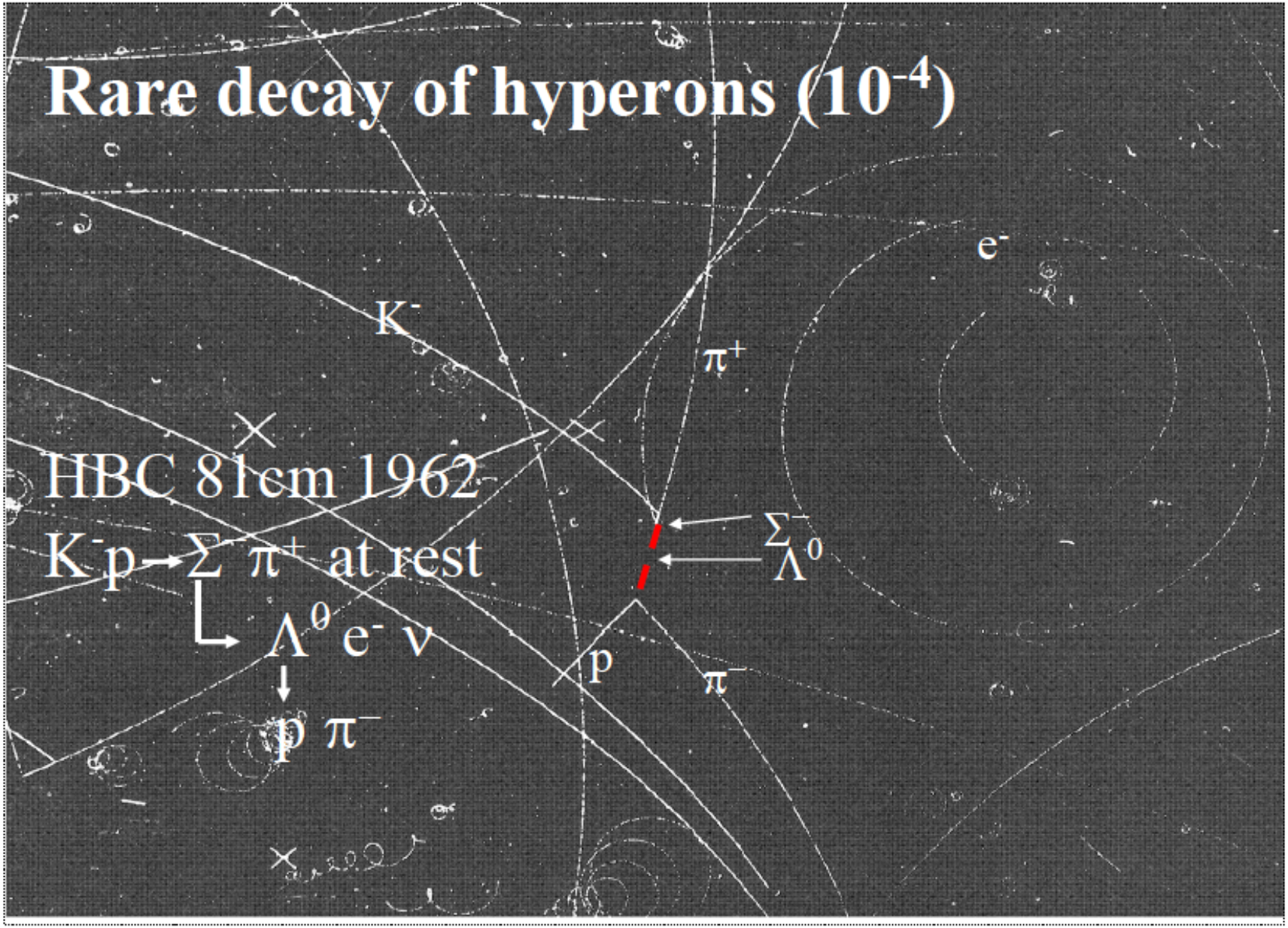
$\vec{B}$

$\otimes$

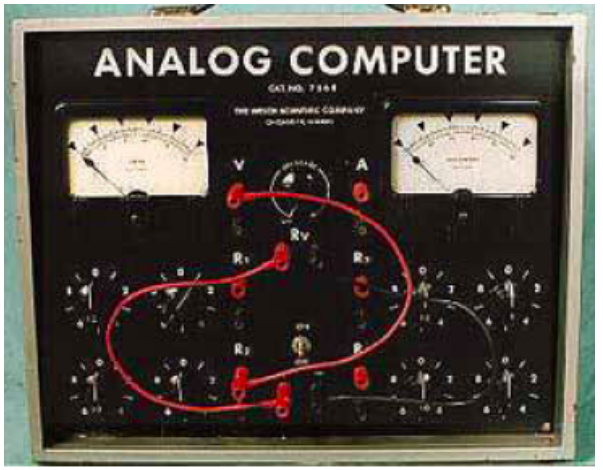
Remember the Lorentz Force

$$\vec{F} = q\vec{v} \wedge \vec{B}$$

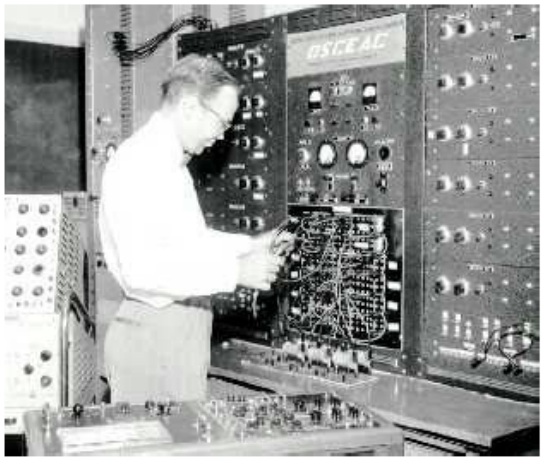



Rare decay of hyperons (10^{-4})
K^-
π^+
e^-
HBC 81cm 1962
$K^- p \rightarrow \Sigma^- \pi^+$ at rest
Σ^-
Λ^0
$\Lambda^0 \rightarrow e^- \nu$
p
π^-
$p \pi^-$

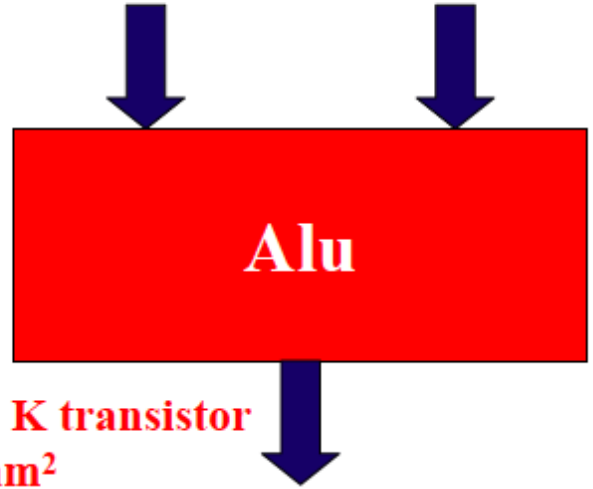
Il trionfo del computing digitale come paradigma di calcolo !?



VS



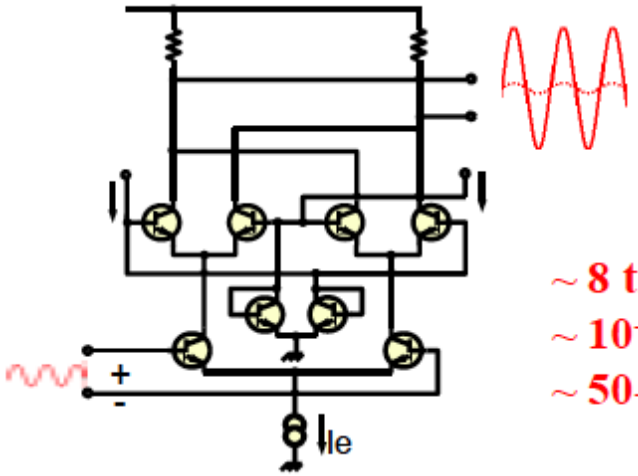
digitale vs analogico



~ 500 K transistor
~ 1 mm²
~ 20-50 ns

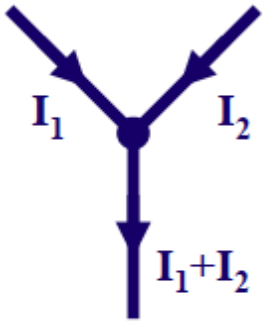
Il digitale non e' piu' veloce e compatto dell'analogico

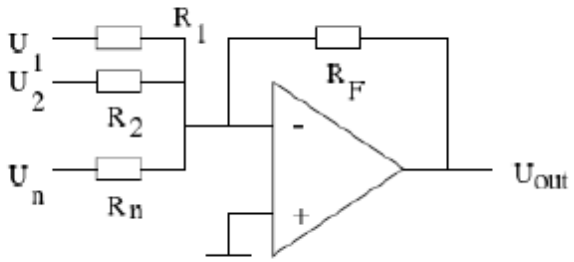
Moltiplicatore di Gilbert



~ 8 transistor
~ 10^{-2} mm²
~ 50-100 ps

Sommatore

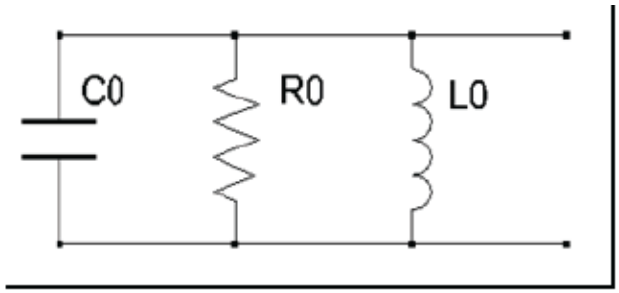




Il digitale non e' piu' preciso dell'analogico

$$\frac{d^2v}{dt^2} = -\frac{1}{RC}\cdot\frac{dv}{dt} - \frac{v}{LC}$$

$$v(t) = 100\cdot e^{-200t}\cdot\sin(979.8t)$$



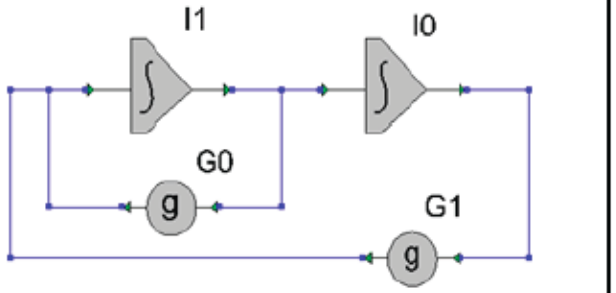
$V_0 = 0$

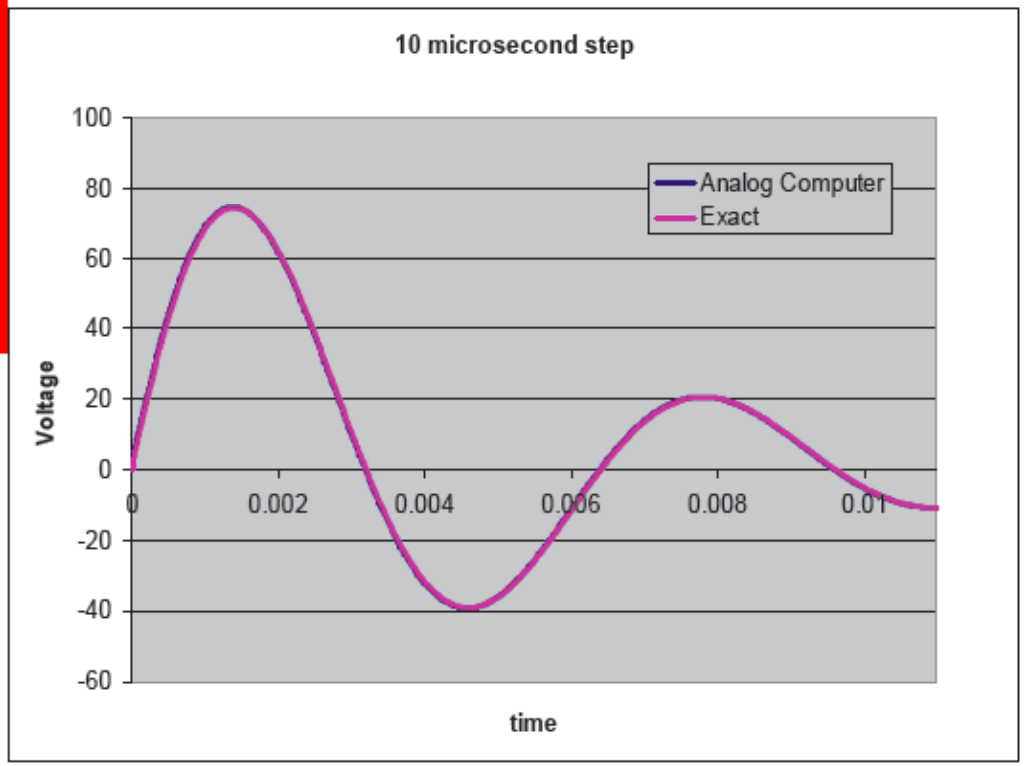
$I_{L_0} = -12.25mA$

$R = 20k\Omega$

$L = 8H$

$C = 0.125\mu F$





The analog computer

The analog computer → any device that 'computes' by means of an analog between real, physical, CONTINUOUS, quantities and some other set of variables.

An analog/analogue computer → is a form of computer that uses electronic or mechanical phenomena to model the problem being solved by using one kind of physical quantity to represent another.

Real quantities → the distance between points on a scale, the angular displacement, the velocity, or the acceleration of a rotating shaft, a quantity of some liquid, the electrical current in a conductor.

Computing analogico

Shannon introduced the General Propose Analog Computer (GPAC) as a mathematical model of an analog device, the Differential Analyzer in 1941. The Differential Analyzer was used from the 30s to the early 60s to solve numerical problems, especially differential equations, for example, in ballistics problems. These devices were first built with mechanical components and later evolved to electronic versions. A GPAC may be seen as a circuit built of interconnected black boxes, whose behavior is given by Fig. 1, where inputs are functions of an independent variable called the time. Shannon's model was extended by others to neural networks and extended Analog computers.

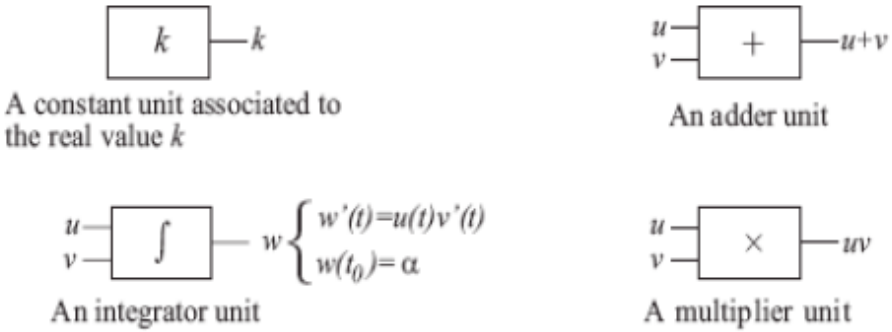
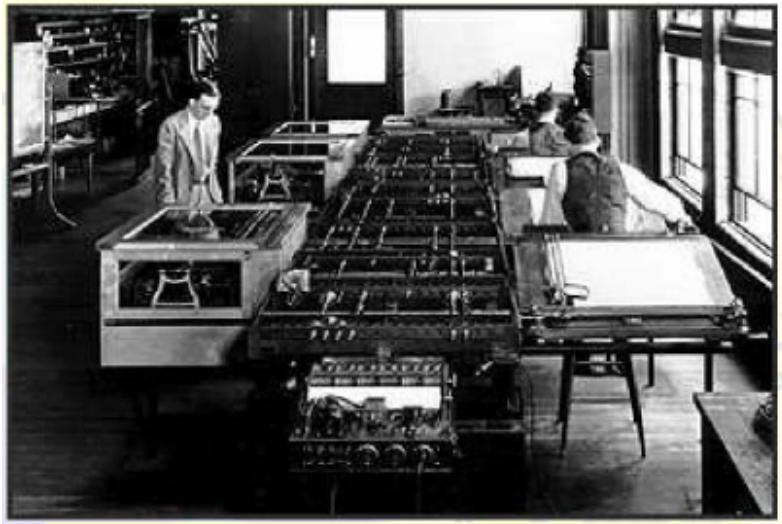


Fig. 1. Different types of units used in a GPAC.

Differential analyzers/GPAC



Vannevar Bush's Differential Analyzer

1876 Thomson first thought of interconnecting mechanical integrators to compute.
1931 A differential was built by Bush at MIT.
1941 Shannon modelized the differential analyzer as a GPAC.

Differential Analyzers

Apart from simple devices like these, more complex instruments have been devised by various scientists like James Thomson (the brother of Lord Kelvin) and for example Vannevar Bush.

At the heart of instruments like these were mechanical devices capable of integration, summing and function generation.

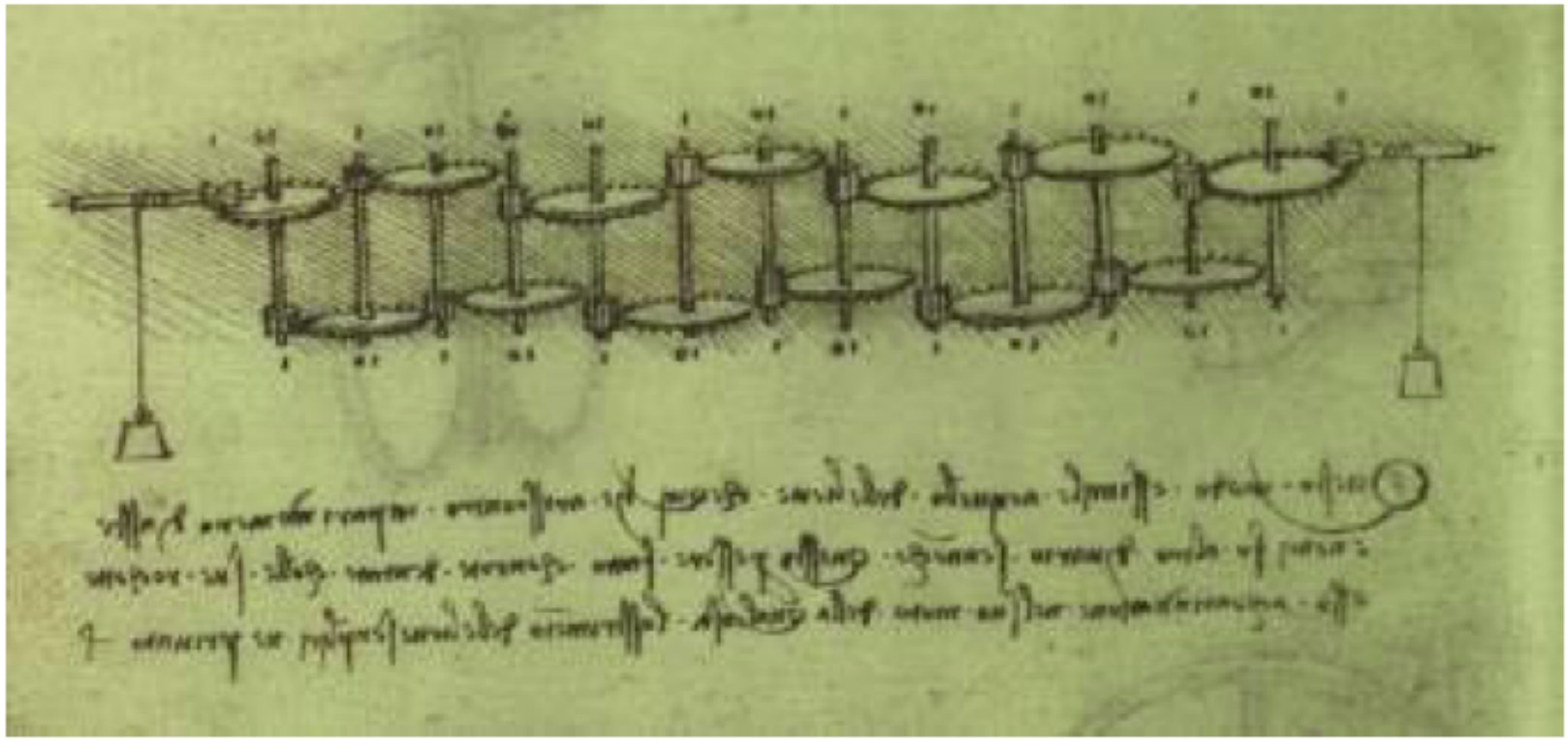
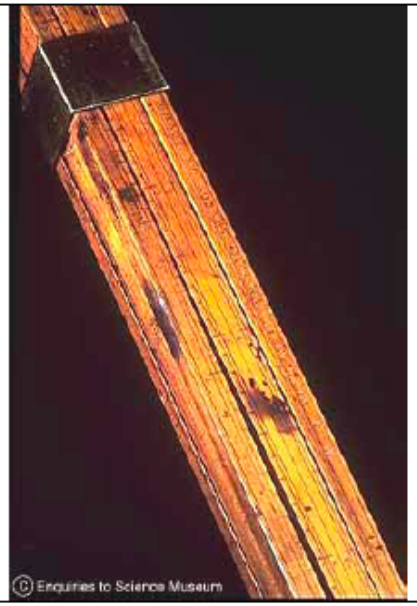
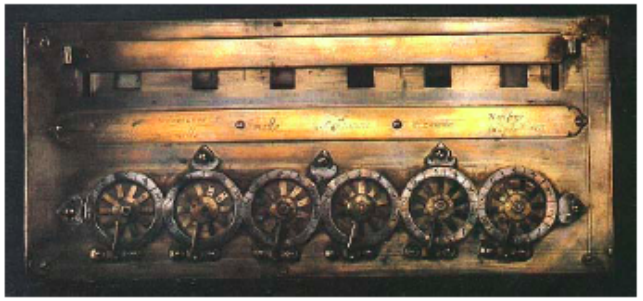
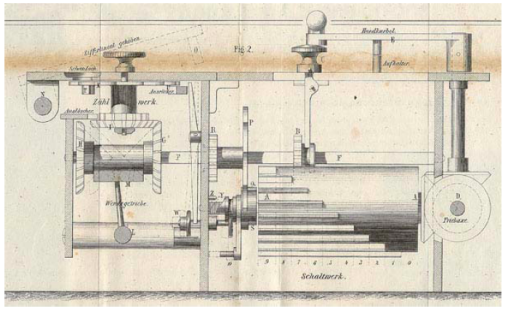


Fig. Una delle macchine disegnate da Leonardo da Vinci: un possibile modello di calcolatore?







- l'invenzione dei ***logaritmi*** e la scoperta delle loro proprietà aprirono la strada all'invenzione del regolo calcolatore (logaritmico), uno degli strumenti più versatili ed apprezzati;

- l'invenzione degli **orologi (a pendolo)** con il conseguente progresso della meccanica di precisione aprirono la strada alla costruzione delle prime calcolatrici meccaniche.

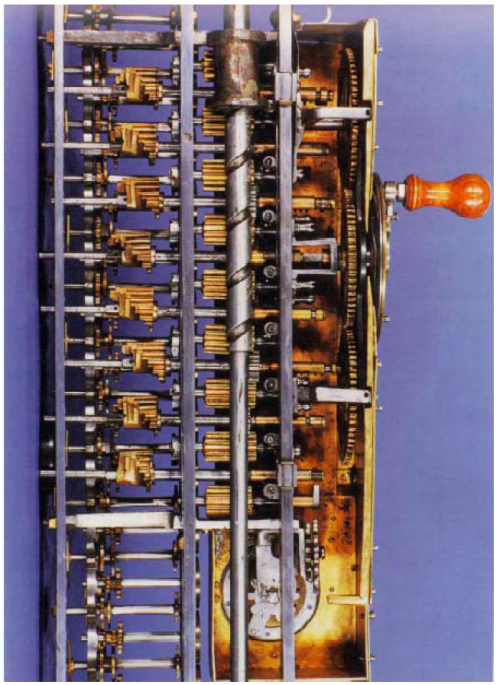


Fig. Interno della calcolatrice di Leibniz.

La ***rappresentazione analogica*** si appoggia al concetto di grandezza fisica continua (ad esempio, la lunghezza o il peso) e rappresenta ogni numero come il risultato della misurazione di tale grandezza su un oggetto specifico (ad esempio, un regolo).

Nella ***rappresentazione numerica*** **(o *digitale*)**, il numero viene rappresentato in modo discreto come un insieme composto da unità elementari considerate indivisibili (ad esempio, in una taglia ogni singola tacca o in un abaco ogni singolo sassolino).

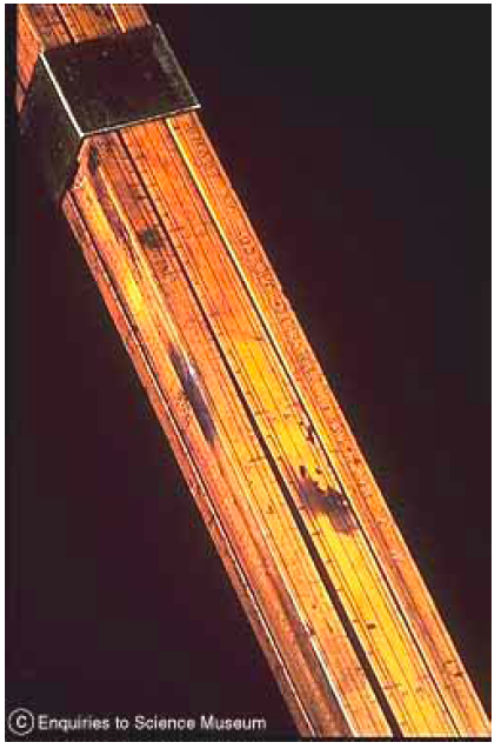


Fig. Regolo di Bissaker (Museo delle Scienze di Londra), uno degli strumenti più antichi a noi giunti, 1654.

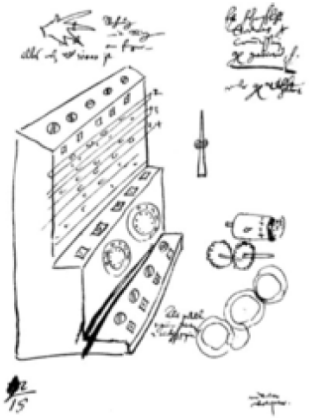
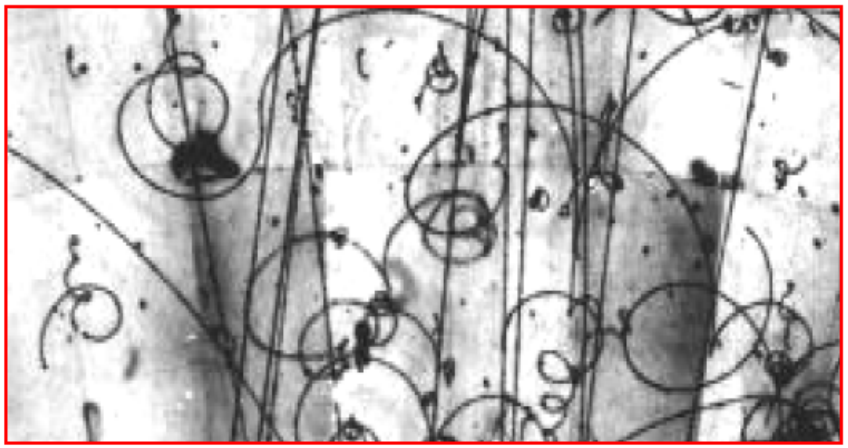


Fig. Ricostruzione dell'Orologio Calcolatore di Schickard (Deutsches Museum, Monaco).

Nell'**approccio analogico** il calcolo avviene sfruttando due aspetti:

- le quantità numeriche sono rappresentate mediante una qualche **grandezza fisica continua**,
- **l'operazione matematica viene simulata con un fenomeno avente un comportamento (cioè una legge fisica) "analogo" all'operazione da effettuare**.

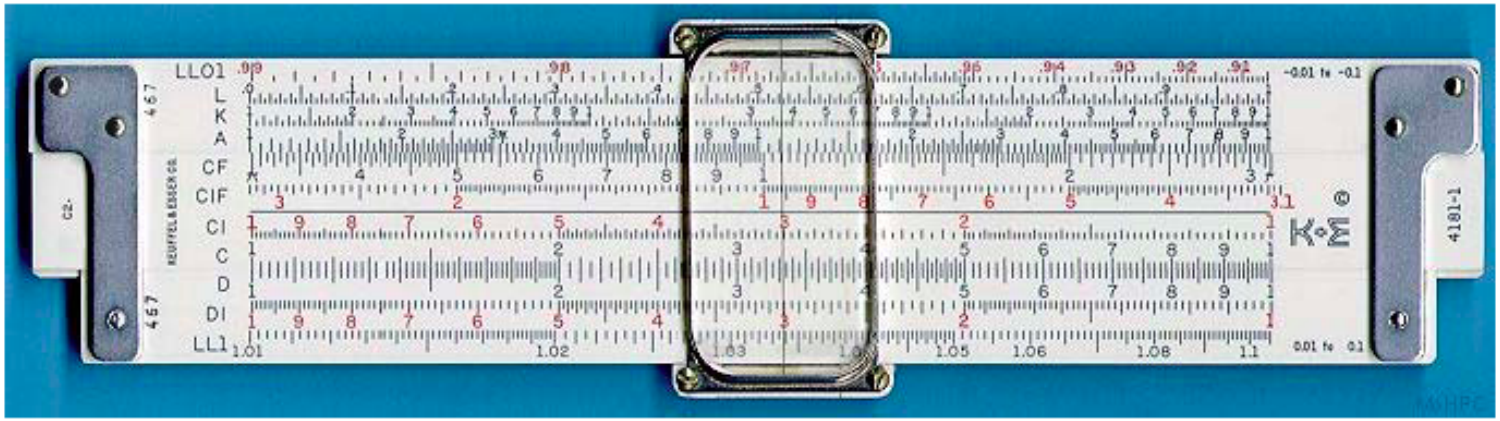


$\vec{B}$

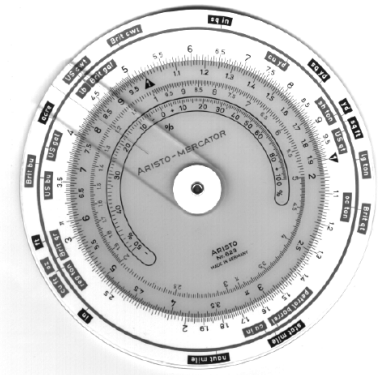
$\otimes$

Remember the Lorentz Force

$$\vec{F} = q\vec{v} \wedge \vec{B}$$



Fino agli anni '70 il regolo calcolatore è stato lo strumento principe degli ingegneri e di tutte le persone impegnate in attività di calcolo scientifico, ed è stato sostituito solo dalle calcolatrici tascabili elettroniche.



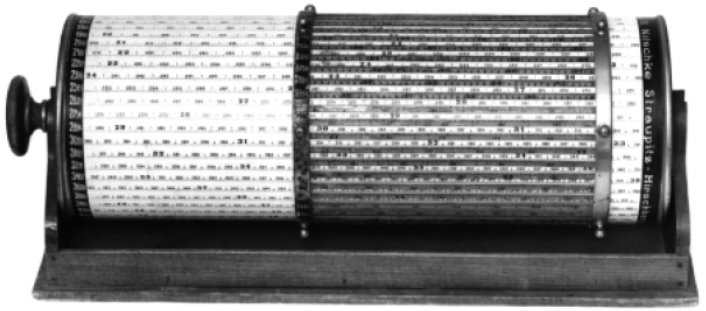
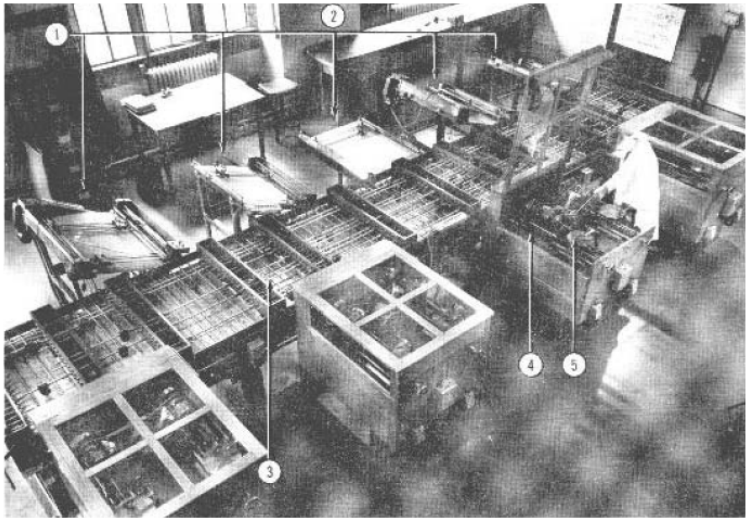
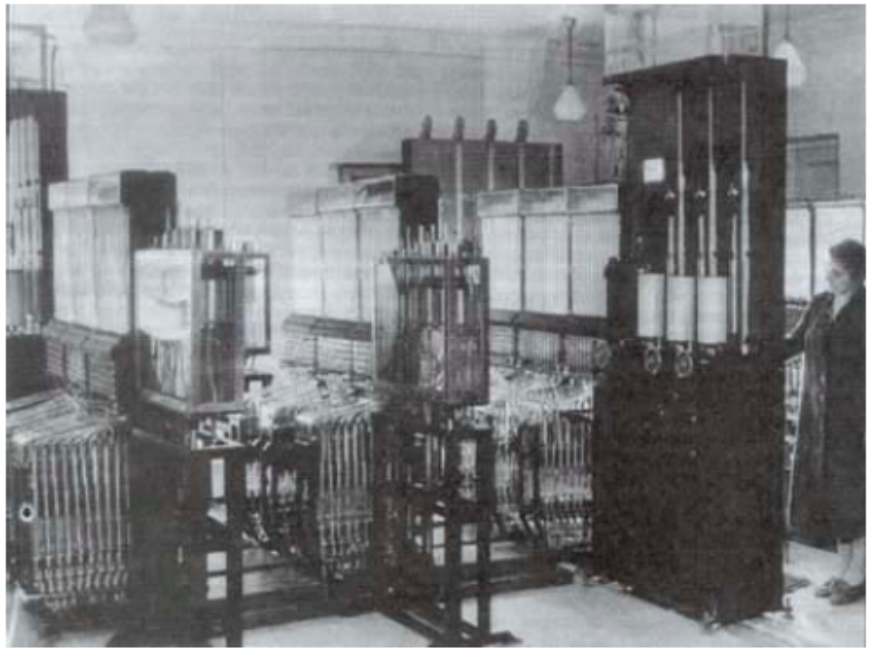


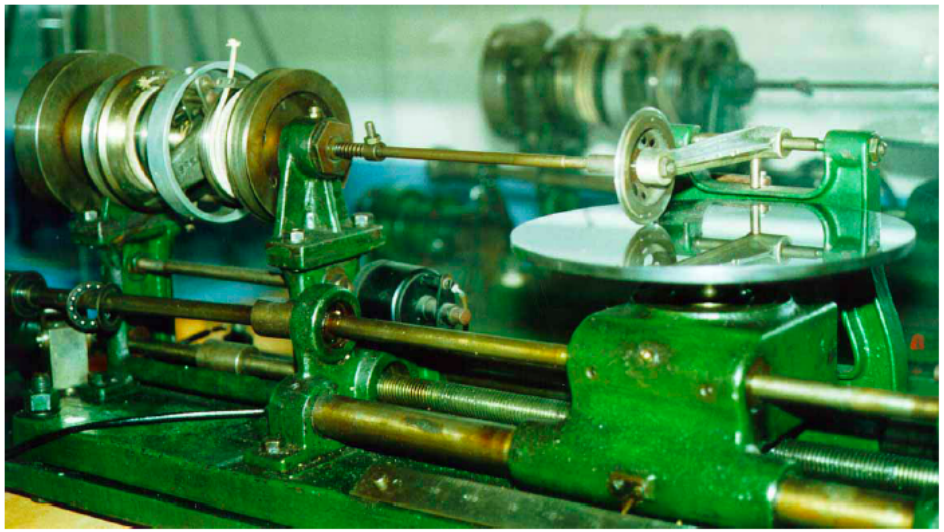
Fig. Regolo calcolatore cilindrico (tipo Nitschke).



Il culmine della ricerca nelle macchine analogiche è costituito dai cosiddetti ***analizzatori differenziali***, dispositivi meccanici in grado di risolvere equazioni molto complesse. Introdotto da **Vannevar Bush** (1890-1974) alla fine degli anni '20, l'analizzatore differenziale era un computer di tipo analogico utilizzato **per risolvere equazioni differenziali** mediante un'integrazione di tipo meccanico.

An early analog computer built in the Soviet Union in 1936. It functioned by careful manipulation of water through a room full of interconnected pipes and pumps. The level of water in various chambers (with precision to fractions of a millimeter) represented stored numbers, and the rate of flow between them represented mathematical operations. Amazingly, this machine was capable of solving non-homogeneous differential equations.





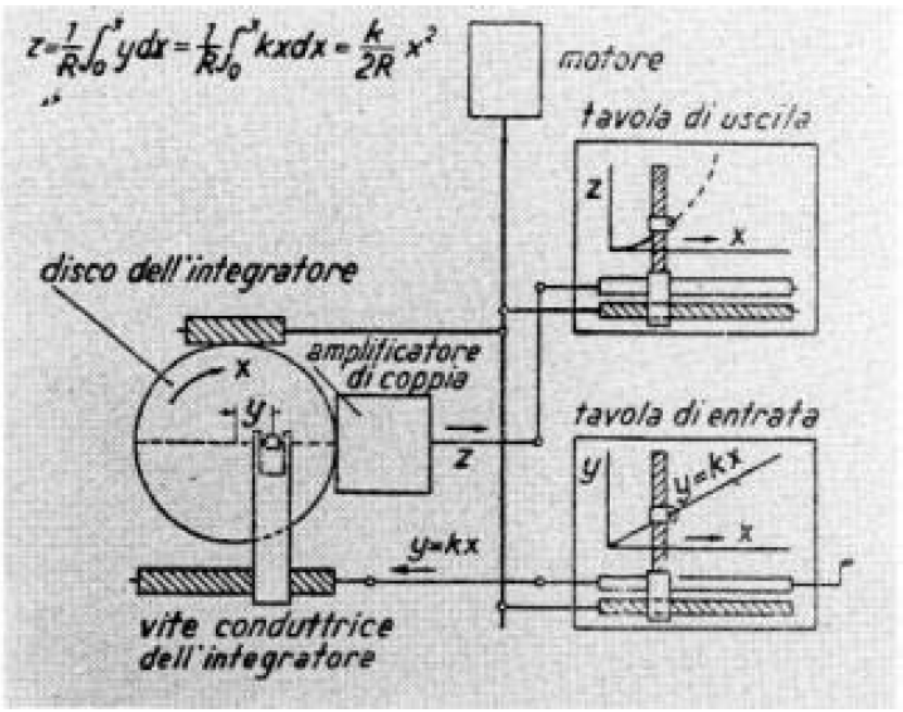
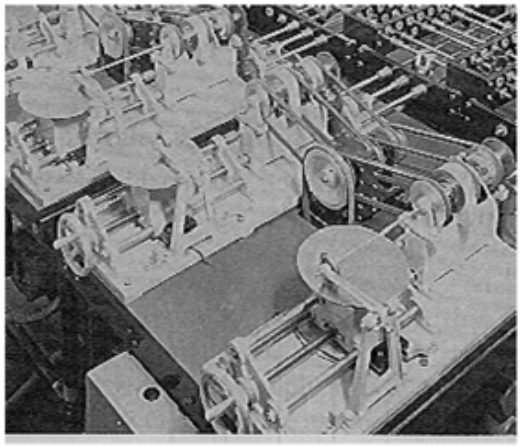
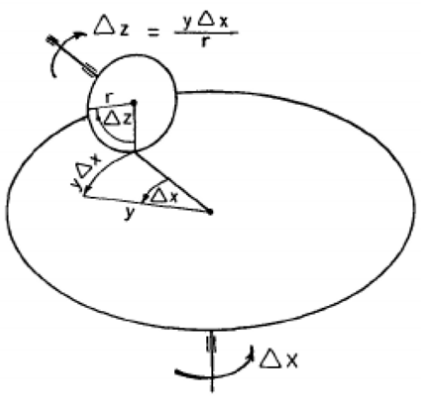
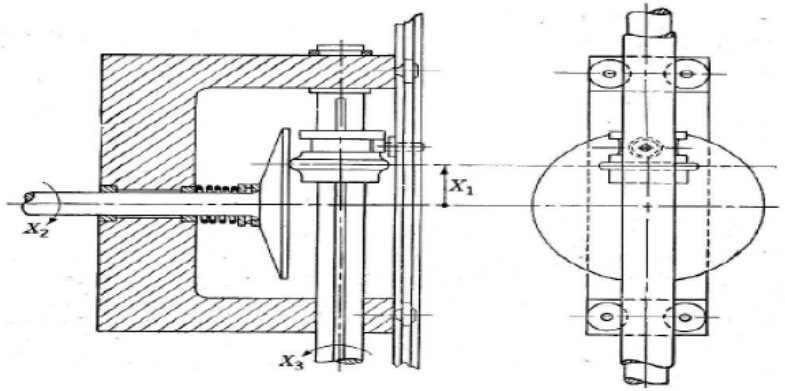


Fig. Vista delle ruote e dischi di integrazione della macchina. I dispositivi che sembrano dei motori sulla sinistra sono gli "amplificatori di torsione di Nieman", dispositivi a guida d'argano che funzionavano come amplificatori meccanici per poter procedere nelle successive operazioni di integrazione senza perdita di accuratezza.

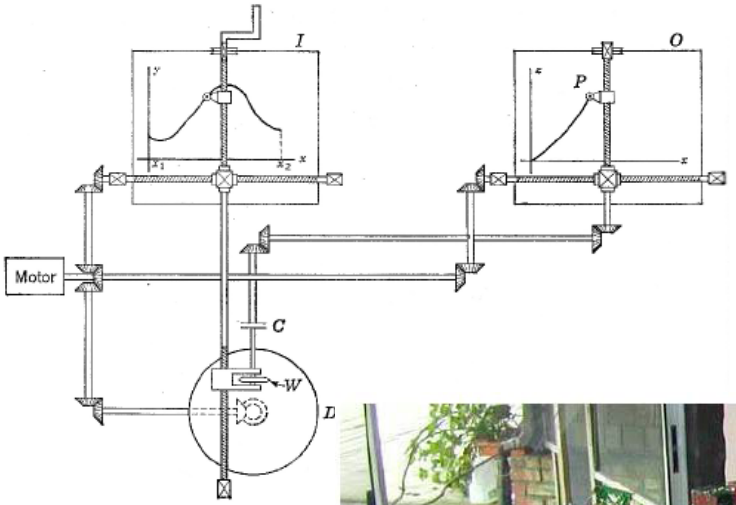


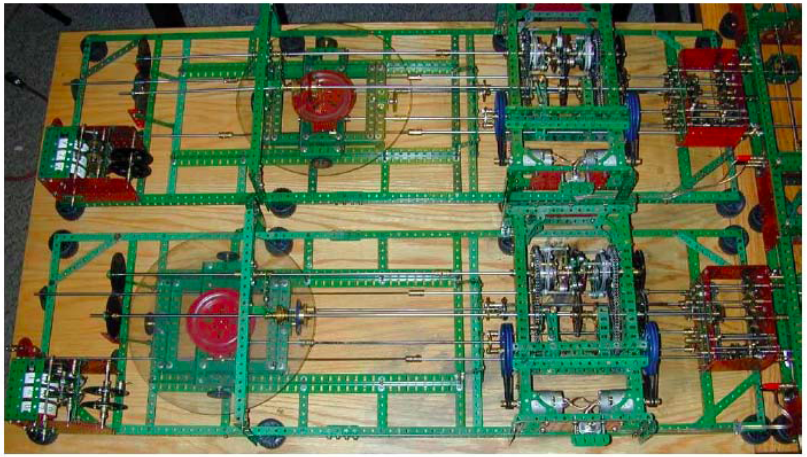


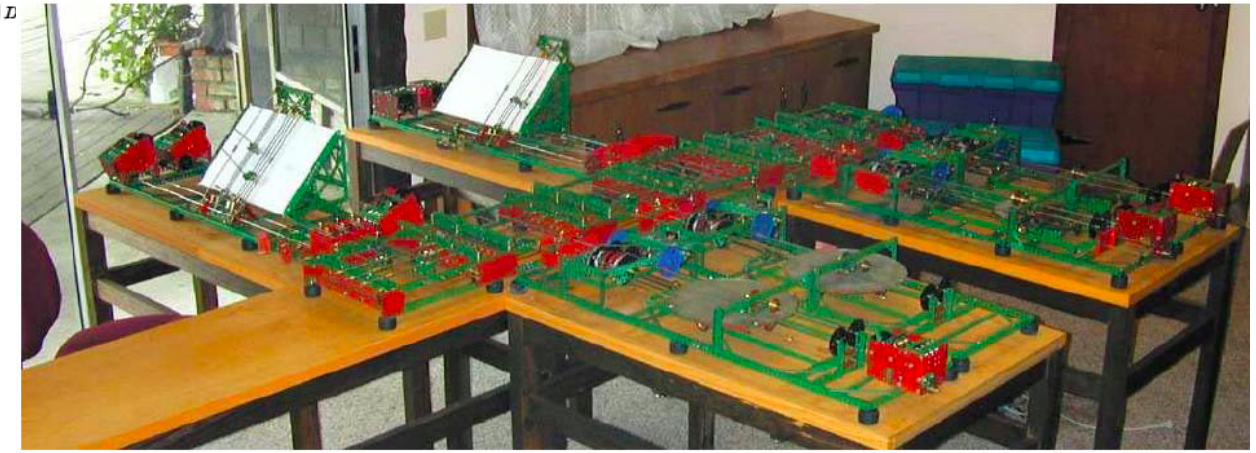


The mechanical differential analyzer

The following picture shows a complete setup of a simple differential analyzer integrating over an arbitrary function (gained from an ”input table”). The result of the integration is plotted on an ”output table” (see [5][p.190]):

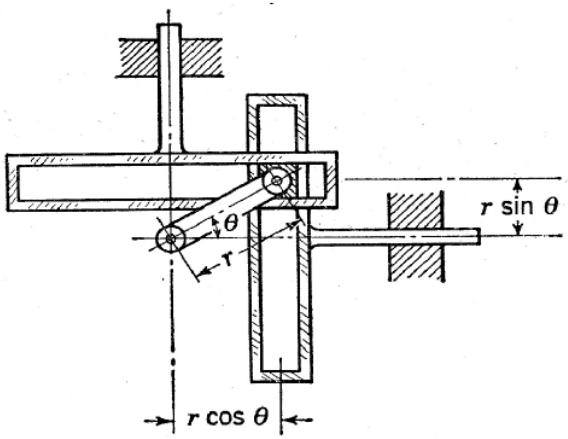






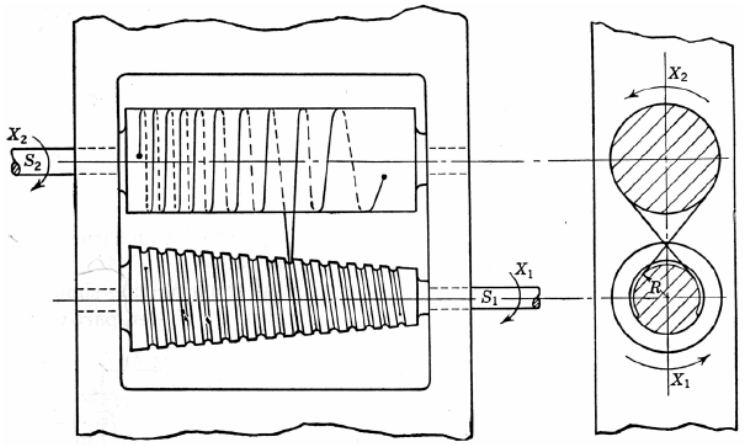
The harmonic synthesizer

Even the generation of trigonometric functions is quite simple using mechanical devices as the following picture shows (see [5][p.242]):



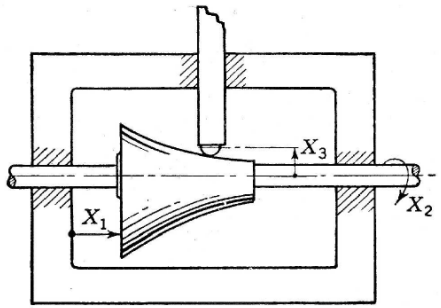
A squaring device

Special functions like $f(x) = x^2$ may be implemented as follows (see [9][p.22]):



Generating $f(x, y)$

Even more complex functions of two variables may be implemented as well (cf. [9][p.24]):



The end of the mechanical era

Mechanical differential analyzers (analog computers) had many disadvantages. This was the reason for their short life time:

- They were large, clumsy and difficult to maintain.
- The speed of calculations is severely limited by the moment of inertia of the components.
- The accuracy is limited to a few percent due to backlash, etc.
- The time required to setup the computer for a specific problem is very high because a lot of mechanical connections must be made.

Electronic analog computers

This led to the development of electronic analog computers which proved to be very influencial and had an incredibly high scientific value.

The first implementation of an electronic analog computer was done by Helmut Hoelzer in the late 1930s and early 1940s during his reserach for the V-weapons of the 3rd Reich (cf. [4][p.202ff]).

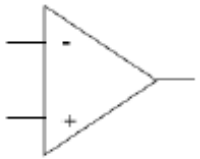
His main interest was in the field of simulation and control of propulsed rocket flight. Controlling a rocket like the V2 required a substantial amount of computational power which required some general approach like the electronic analog computer.

The key element of an all electronic analog computer is the so called *operational amplifier*.

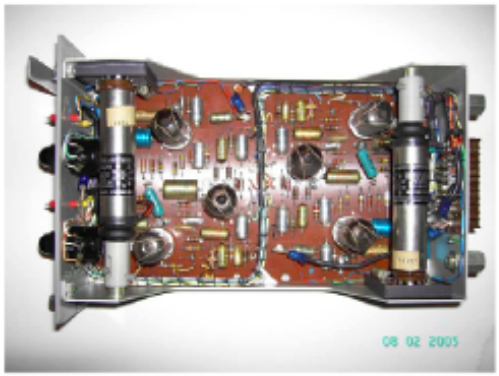
The ideal operational amplifier

In the following an idealized operational amplifier with the following characteristics will be assumed:

- Infinite input impedance.
- Infinite gain.
- No drift.
- Differential inputs.



A real tube based operational amplifier



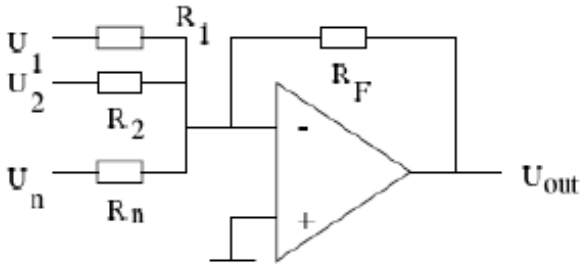
A real operational amplifier

All real implementations of operational amplifiers will suffer from drift, non-zero input impedance and – most important – non-infinite gain. A gain less than ∞ will yield (static) errors in computations so an amplifier suitable for an electronic analog computer will need a very large gain.

The following picture shows a tube based dual amplifier as used in the Solartron Analogue Tutor.

Both amplifiers are chopper stabilized to minimize drift effects and have a DC gain of 10^9! A most respective value even for today's standards.

Summing



The sum of the currents at the summing junction has to be 0 due to the assumed gain $G = \infty$:

$$\sum_{i=1}^{n} \frac{U_i}{R_i} = -\frac{U_{\text{out}}}{R_F}$$

If $a_i = \frac{R_F}{R_i}$ then

$$-U_{\text{out}} = \sum_{i=1}^{n} a_i U_i$$

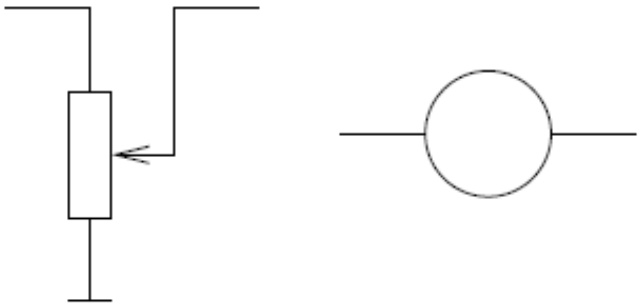
holds. Please note that a summer always implies a change of sign of the result!

Coefficient potentiometers

Nearly every calculation needs some way to multiply a value by a fixed amount.

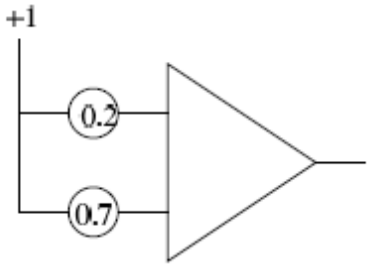
In an electronic analog computer this is normally done with a precision 10-turn potentiometer connected as a voltage divider. This allows multiplication of an input value with a factor $1 \leq a \leq 1$.

The circuit of such a coefficient potentiometer and its graphic representation (a circle in which normally the multiplier is shown) can be seen in the figure below:



A summing example

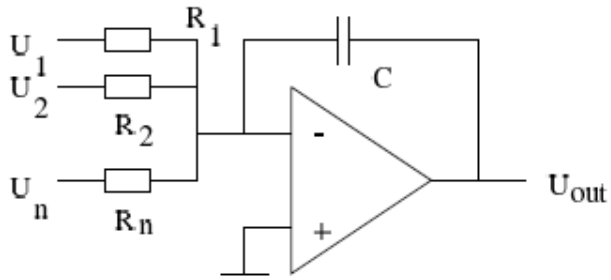
To compute the sum $-(7+2)$ one would setup the analog computer as follows:



The triangle symbol denotes a summer as shown on the slide before, not a simple operational amplifier!

The circles denote two coefficient potentiometers.

Integrating



Integration is just as easy as summing – the main difference between a summer and an integrator is that the latter uses a capacitor as its feedback impedance as shown above. The output voltage of the integrator shown above is

$$-U_{\text{out}}(t) = U_0 + \int_0^t \sum_{i=1}^{n} \frac{1}{R_i C} U_i \, \mathrm{d}t$$

where U_0 is the so called *initial condition*. Just like a simple summer an integrator changes the sign of its output.

Integrator control

The integrator is not only at the heart of every analog computer, it is by far the most complicated device since its behaviour does not only depend on its (analog) inputs.

An integrator can be run in ony out of three modes:

Mode	Description
Initial condition	In this mode the input summing network is disconnected from the amplifier while another input path is connected to the integration capacitor to charge it to the desired initial value.
Compute	The integrator calculates the time integral.
Halt	The integrator stops the calculation and holds the current value.

Run modes

An electronic analog computer normally features several run modes with respect to the control of its integrators:

Run mode	Description
Initial condition	All integrators are switched to IC.
Run	All integrators are placed in run mode.
Halt	All integrators enter the hold state.
Run with periodic halt	IC, Run, Hold, Run, Hold, Run, Hold, . . .
Repetitive run	IC, Run, IC, Run, IC, Run, . . .
Iterative run	The integrators are pooled into (at least) two groups: A normal group and a complementary group. While one group is in Hold the other one is in IC and vice versa.

What about deriving?

Obviously interchanging the feedback capacitor and the input resistor(s) would change the integrator into a differentiator which would seem like a good idea given the fact that differential equations rely on deriving instead of integrating.

Nevertheless analog computers normally make use of integrators only. This is a result of the fact that an integrator smoothes a signal thus suppressing noise, while a differentiator amplifies noise.

This requires one to get rid of all derivatives in a differential equation by integrating it a number of times before solving the equation(s) on an analog computer.

Solving a simple differential equation

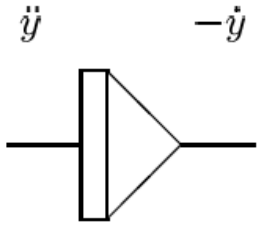
Using summers and integrators it is now possible to solve simple differential equations like

$$\ddot{y} = -y.$$

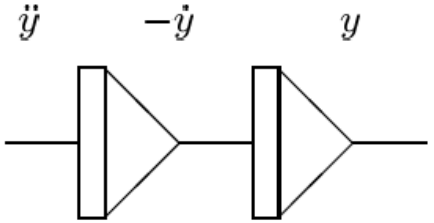
First let us assume that $\ddot{y}$ is known...

Solving a simple differential equation

Using $\ddot{y}$ as the input to an integrator we get the negative of its integral:

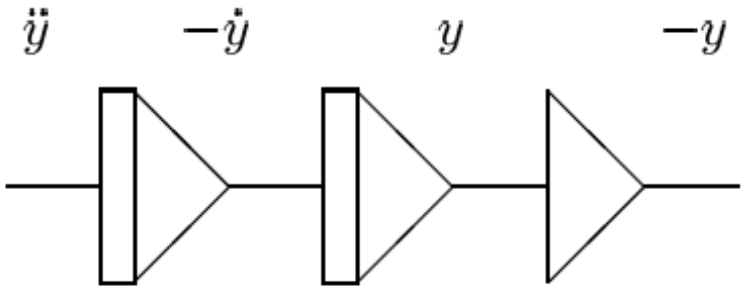


Applying another integrator yields something quite familiar:

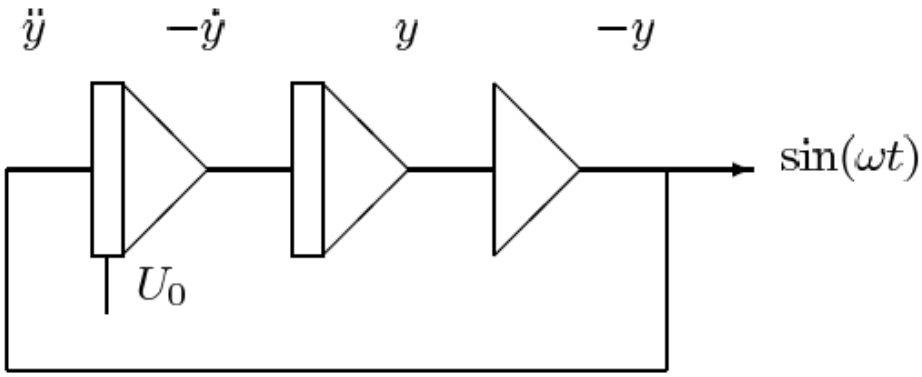


Solving a simple differential equation

Using a summer with one input the sign of the result can be changed easily:

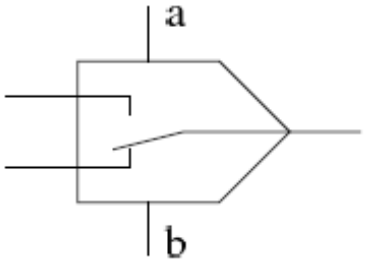


Obviously the left hand side has to equal the right hand side according to the original differential equation, so connecting both sides realizes the equation on the analog computer yielding $\sin(\omega t)$ as the output of the computer circuit (U_0 denotes the initial condition, usually 1 in this case):



Comparators

If a high gain operational amplifier with a power output stage is used to drive a relay (or better an electronic switch), the resulting device is normally called a *comparator*. Its symbol is shown below:



Such a comparator normally has two control inputs a and b. It calculates the sum $a + b$. If this sum is greater than zero, the relay will be placed in the upper position, otherwise it will be in the lower position.

Using a comparator makes it possible to change the computing circuit during a calculation depending on external events such as reaching a boundary, etc.

Multiplication

Apart from summers, integrators and coefficient potentiometers, most relevant differential equations require the use of multipliers to solve them.

Multiplication is by no means simple using analog electronic circuits only, and a number of different approaches have been implemented:

- Servo multipliers
- Electron beam multipliers
- Hyperbolic field multipliers
- Time division multipliers
- Parabola multipliers
- Flux compensation multipliers (just kidding).

Servo multipliers

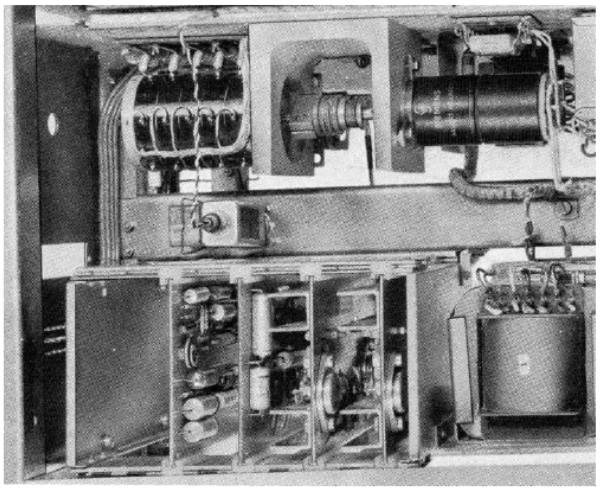
The earliest scheme adopted to perform electromechanic multiplication is the so called *servo multiplier*. It is a closed loop servo mechanism with a motor driven feedback potentiometer.

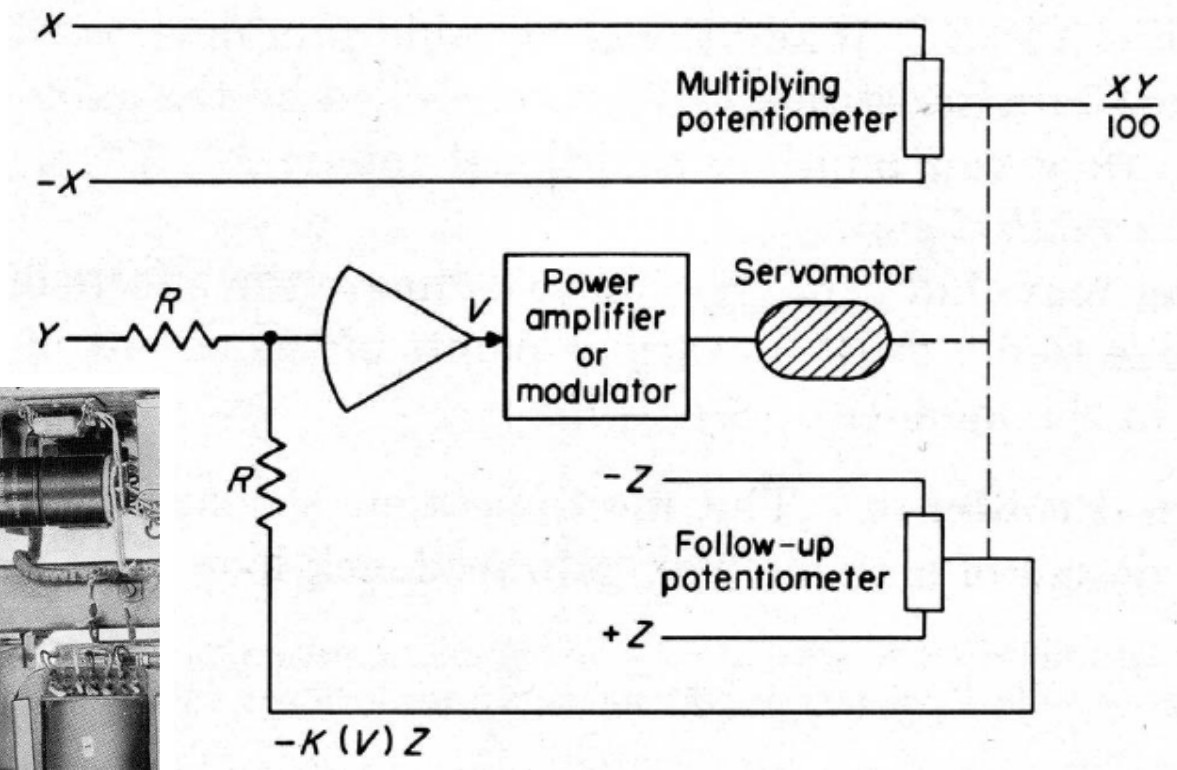
The position of the slider of this feedback potentiometer is controlled by an input signal, the multiplier. An operational amplifier with a power output stage capable of driving the servo motor will always set the angular position of the potentiometer in such a way that it corresponds to the input signal.

Some additional potentiometers are mounted on the same shaft as the feedback potentiometer. These potentiometers are used to perform multiple multiplications of different multiplicands by a single (and common) multiplier (which is the input signal to the servo loop).

Servo multipliers

The following picture shows the simplified schematics of a servo multiplier (see [7][p.260]):

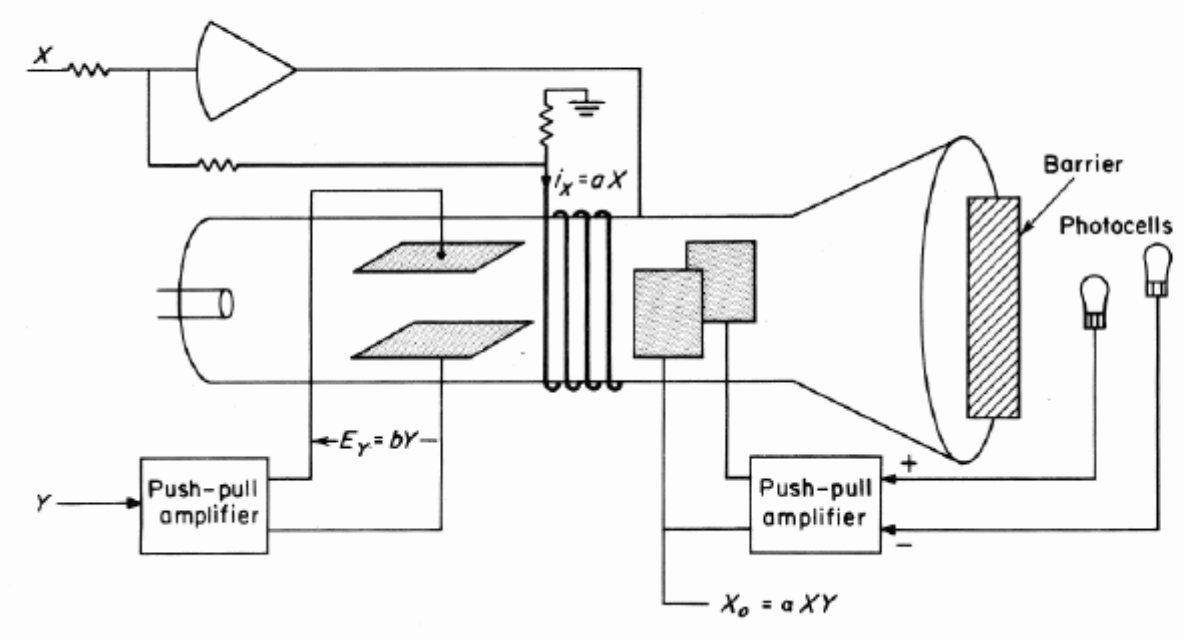




Electron beam multipliers

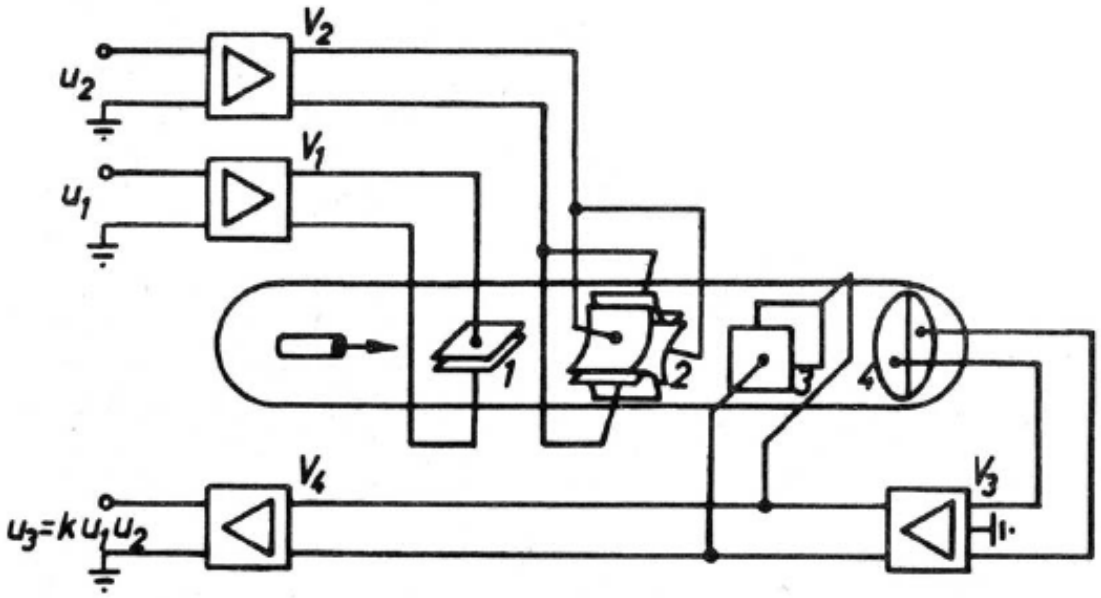
Another multiplication technique makes use of electrostatic and electromagnetic deflection of an electron beam in an oscilloscope tube. The bandwidth of a device like this is limited by the magnetic deflection of the beam only.

The following picture shows an electron beam multiplier (cf. [7][p.294]):



Hyperbolic field multipliers

To overcome this bandwidth limitation, the hyperbolic field multiplier was developed in which only electrostatic deflection is used. Unfortunately this requires complex formed deflection plates and limits the obtainable precision of the device (see [1][p.17]):



Parabola multipliers

A completely different multiplication technique is based on the fact that

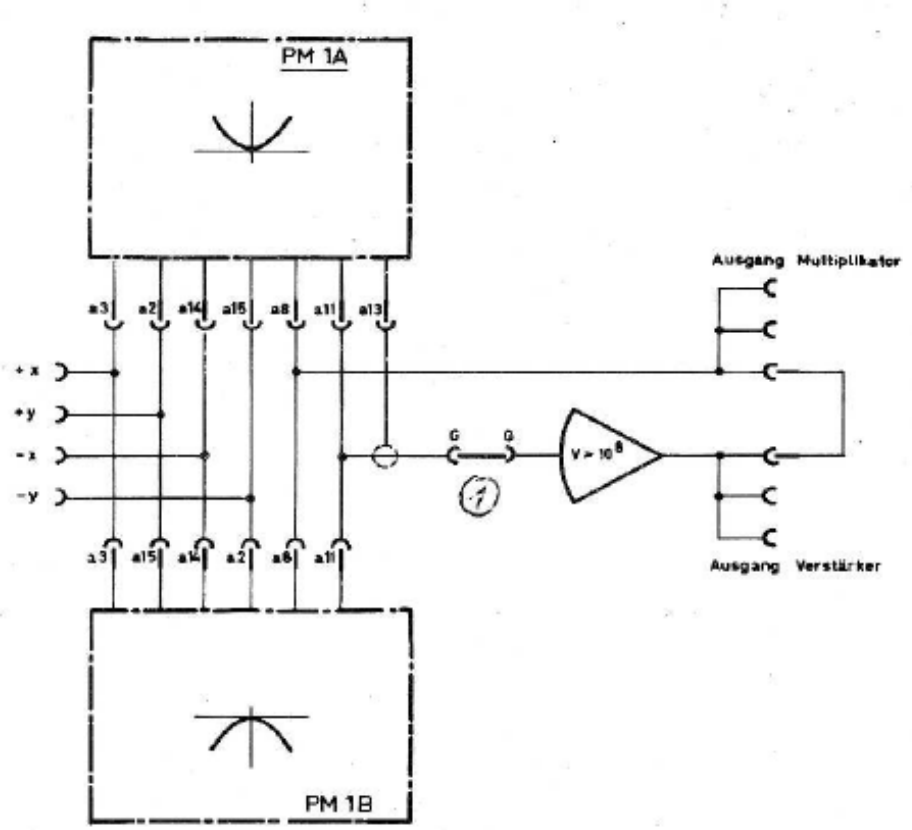
$$\frac{1}{4}(x+y)^2 - (x-y)^2 = xy$$

The generation of the two square functions is performed by diode function generators which approximate a function by polygons. This limits the obtainable precision of a parabola multiplier to some 10^{-4} which is enough for nearly all applications.

The main advantage of such a multiplier is its very large bandwidth and its simple implementation. It requires only some biased diodes and an operational amplifier. A disadvantage is that both variables to be multiplied have to be available with both signs!

Parabola multipliers

The following picture shows the schematics of such a parabola multiplier as it is used in most transistorized analog computers (see [10][p.26]):

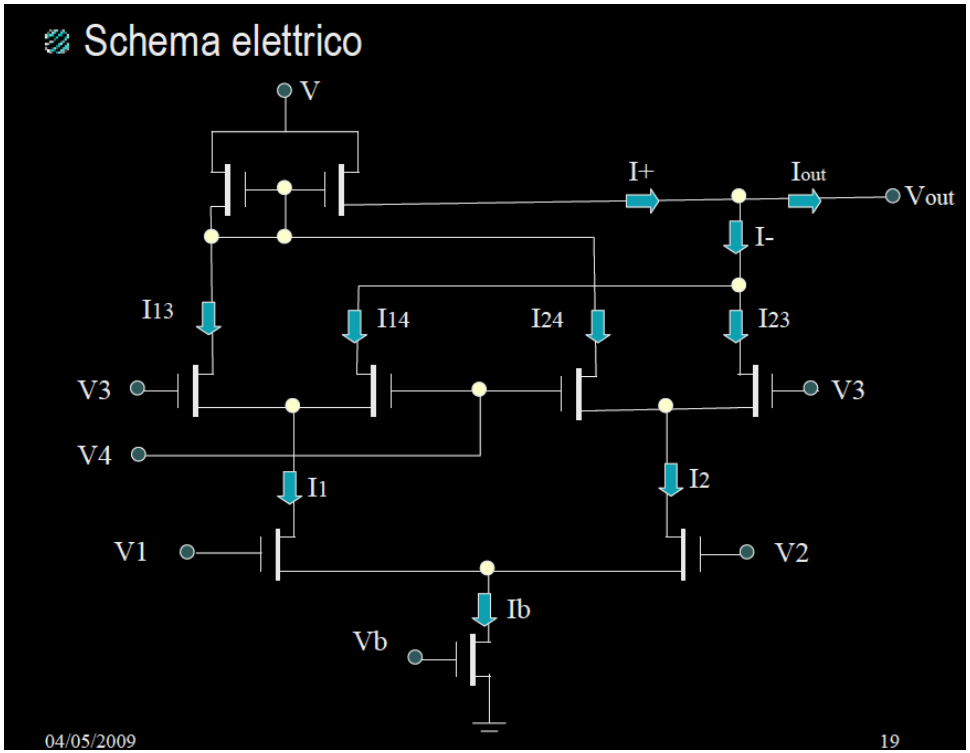


Moltiplicatore a quattro quadranti

- Moltiplicatore di Gilbert

Tale moltiplicatore viene usato quando si vogliono moltiplicare segnali che possono assumere entrambi i segni, positivo o negativo. In uscita avremo la differenza di due tensioni moltiplicata per la differenza di altre due tensioni.
Vedremo che per piccoli segnali si possono fare delle approssimazioni e noteremo che il moltiplicatore a quattro quadranti altro non è che un caso particolare dell'amplificatore a transconduttanza , infatti viene definito più propriamente come :
moltiplicatore di Gilbert a transconduttanza.

- Schema elettrico



$$I_{out} = I_b \tanh \frac{k\,(V_1 - V_2)}{2} \tanh \frac{k\,(V_3 - V_4)}{2}$$

Per piccoli segnali di ingresso < di kT/q la tanh(x) viene approssimata con (x) , e il circuito si comporta da vero moltiplicatore

Function generators

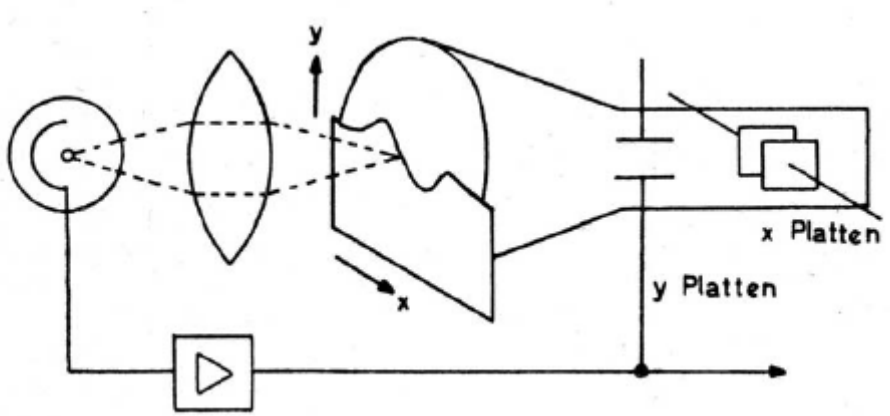
Generating arbitrary functions is of prime importance for analog computers. During the years a wide variety of different function generators have been developed.

These function generators may be grouped like this:

- Function generators based on electromechanical devices like tapped potentiometers in a servo feedback loop (quite like a servo multiplier).
- Function generators based on special physical properties of some substances (voltage dependent resistors, temperature dependent resistors, special semiconductors, etc.).
- Function generators making use of a cathode ray tube in the feedback loop of an operational amplifier.
- Diode function generators which approximate (nearly) arbitrary functions by polygons. These are implemented using biased diodes.

Curve tracing function generators

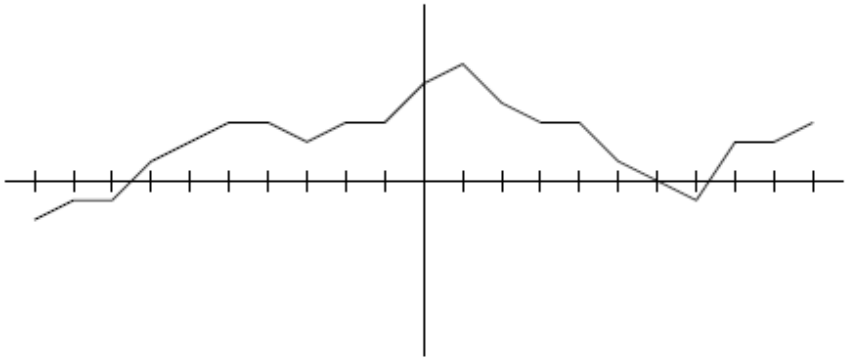
The following picture (see [3][p.A3]) shows an oscilloscope tube based curve following function generator:



The input to the X-deflection plates determines the position at which the function value has to be determined. The function is implemented as an opaque mask placed in front of the tube screen. A feedback loop will generate an Y-deflection voltage in such a way that the beam will just be on top of the mask, thus following the function.

Diode based function generators

Diode based function generators approximate (more or less) arbitrary functions by polygons as shown in the picture below:

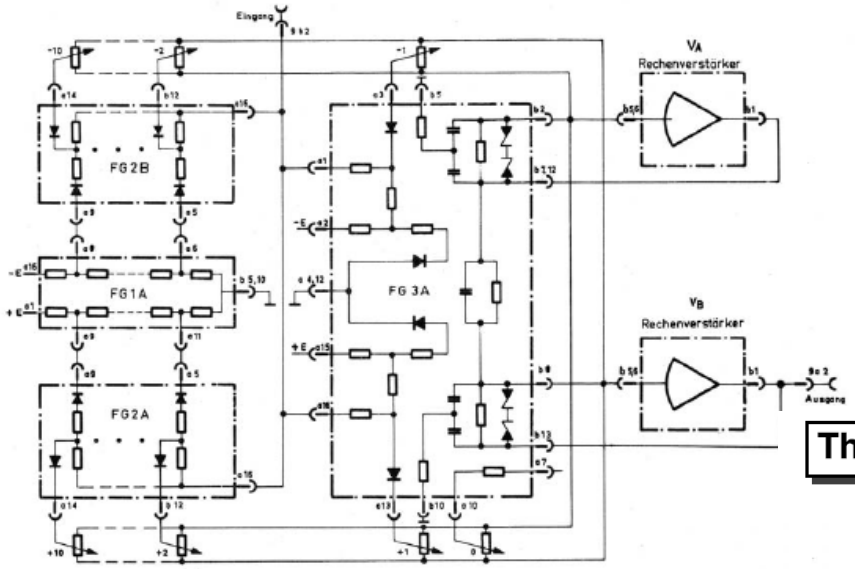


Most implementations have either fixed sampling points (in this case there are normally 21 of these points spaced $\frac{1}{10}E$ apart where E denotes the machine unit (either 10 V or 100 V)) or variable sampling points.

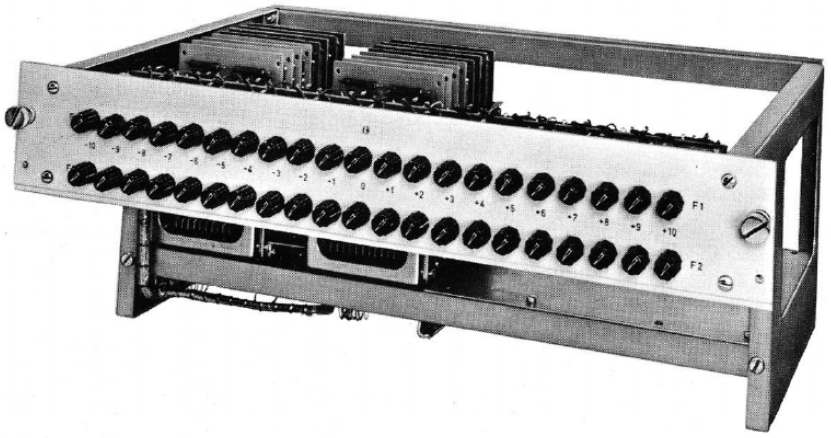
Each sampling point corresponds to a diode biased in a way that it will begin to conduct when the x-input reaches this value. By means of a potentiometer the slope added to the polygon by this diode can be set.

Diode based function generators

The following picture shows the wiring diagram of a diode based function generator allowing positive and negative slopes for the partial polygons (cf. [10][p.28]):

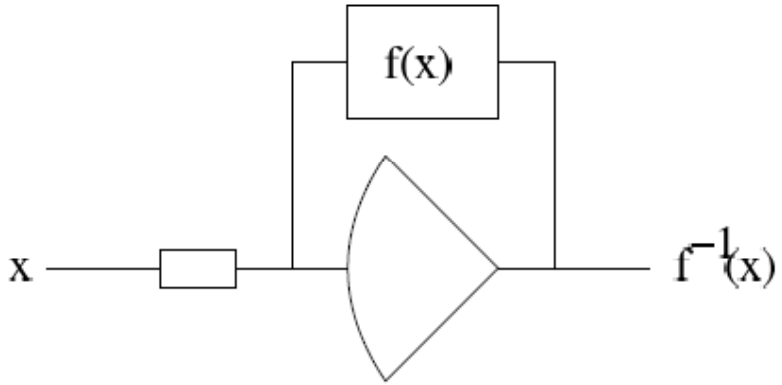


The diode function generator of the RAT700



Generating inverse functions

Generating the inverse $f^{-1}(x)$ of a given function $f(x)$ is quite simple by placing a function generator set up to generate $f(x)$ in the feedback loop of an operational amplifier:

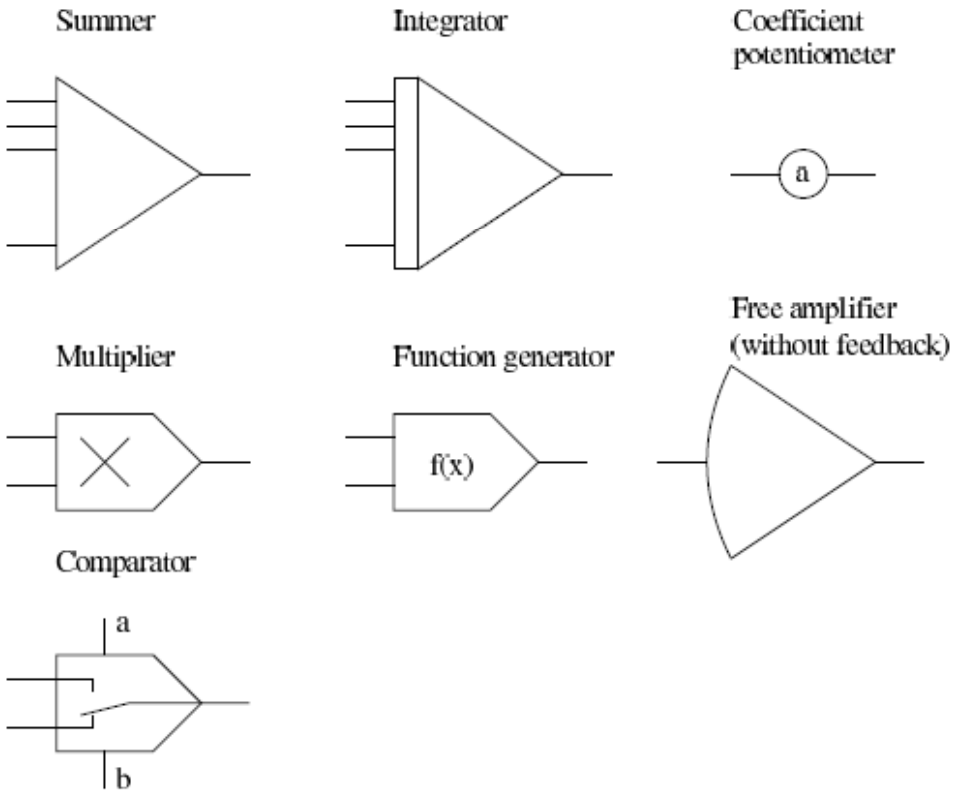


This circuit is normally employed to generate a square root from a square, an exponential function from a logarithmic function, etc.

A practical implementation will normally need a small (some pF) capacitor accross the function generator to enhance the stability of the circuit.

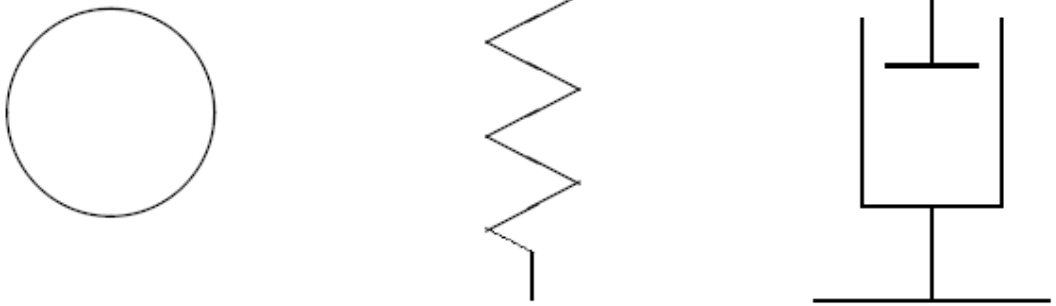
Symbols used in analog computing

The following symbols are normally used in analog computing to denote the various computer elements used in solving a given problem:



Simulating a mass-spring-damper system

The first example shows how to simulate the behaviour of a rather simple mechanical system consisting of a mass, a spring and a damper. The basic elements of this system are shown below with the mathematical representation of the forces belonging to each:



$$F_m = ma = m\ddot{y} \qquad F_s = sy \qquad F_d = d\dot{y}$$

Rearranging the equation

To solve the equation

$$m\ddot{y} + d\dot{y} + sy = 0$$

on an analog computer it is rearranged in a way that yields the highest derivative of y on the left hand side:

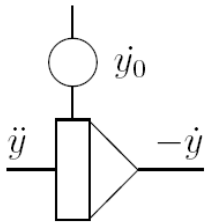
$$\ddot{y} = -\frac{1}{m}\left(d\dot{y} + sy\right).$$

For setting up the computer assume that $\ddot{y}$ is known and generate the remaining terms incorporating lower derivatives of y by successive integration, multiplication and summing of terms.

Note that each summer and each integrator will change the sign!

Generating $-\dot{y}$

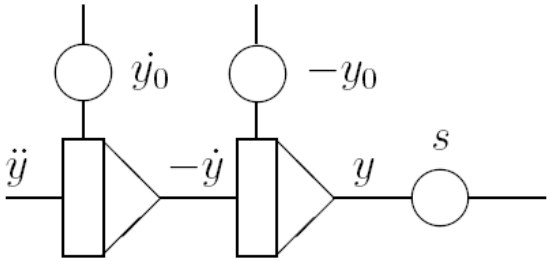
Assuming that $\ddot{y}$ is known, its next lower derivative, $-\dot{y}$, can be generated by using an integrator. The initial condition input of this integrator is used to set the initial value $\dot{y}_0$ as shown in the picture below:



$$\ddot{y} = -\frac{1}{m}\left(d\dot{y} + sy\right).$$

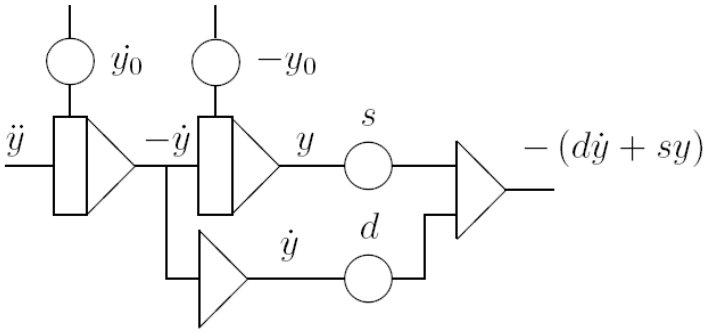
Generating $F_s = sy$

In the following step the force F_s exerted by the spring will be generated:



Generating F_d and the sum of forces

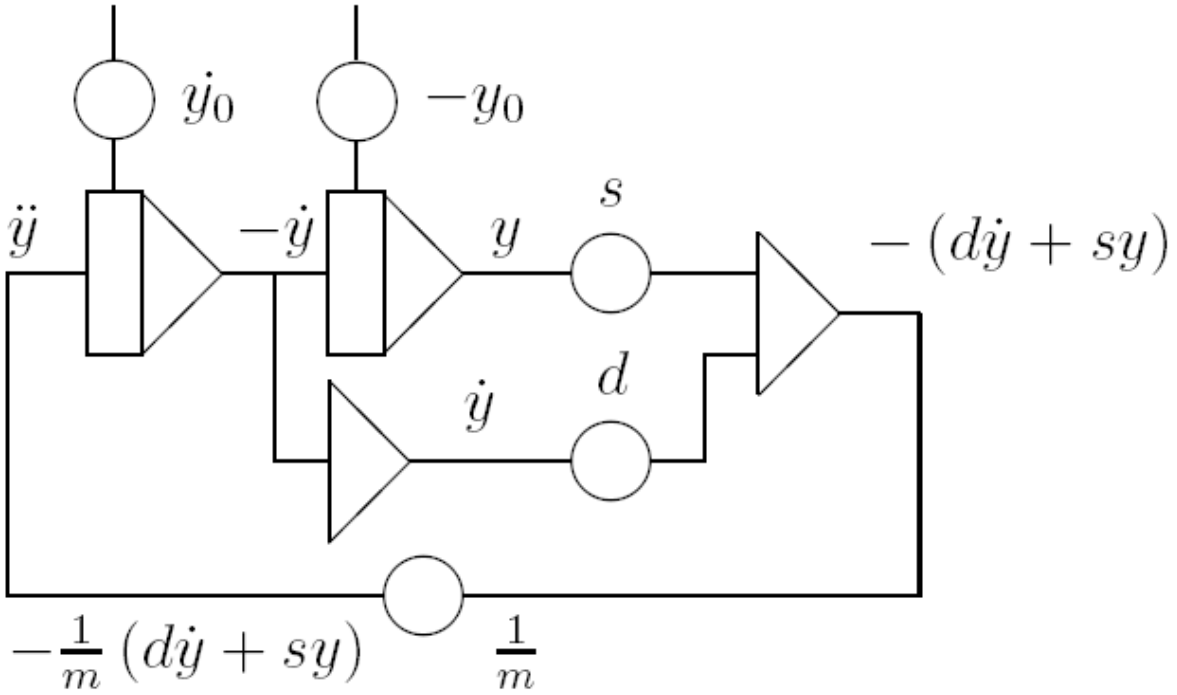
The force generated by the damper, F_d, can be generated accordingly using the already knwon value $-\dot{y}$. The setup shown below then creates the sum of F_s and F_d with a negative sign:



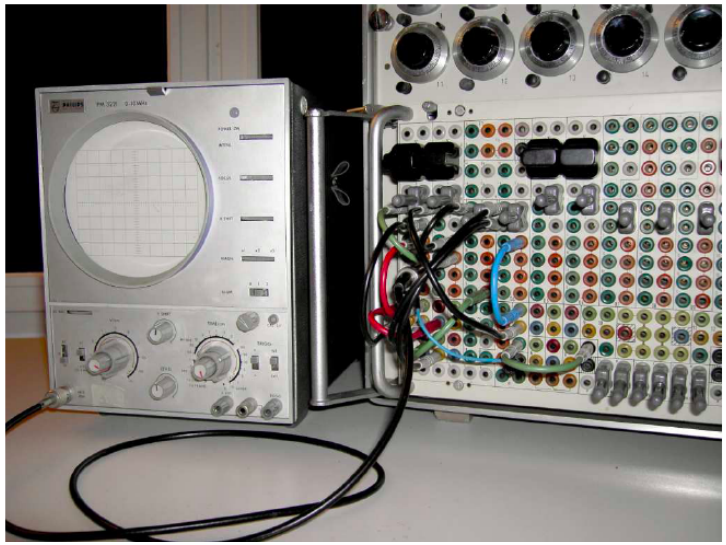
$$\ddot{y} = -\frac{1}{m}\left(d\dot{y} + sy\right).$$

Closing the loop

The sum $-\left(F_s + F_d\right)$ can now be multiplied by the constant $\frac{1}{m}$ yielding $\ddot{y}$ which is exactly what we expected at the input of the circuit. So closing the loop will result in a computer setup solving the initial differential equation readily:

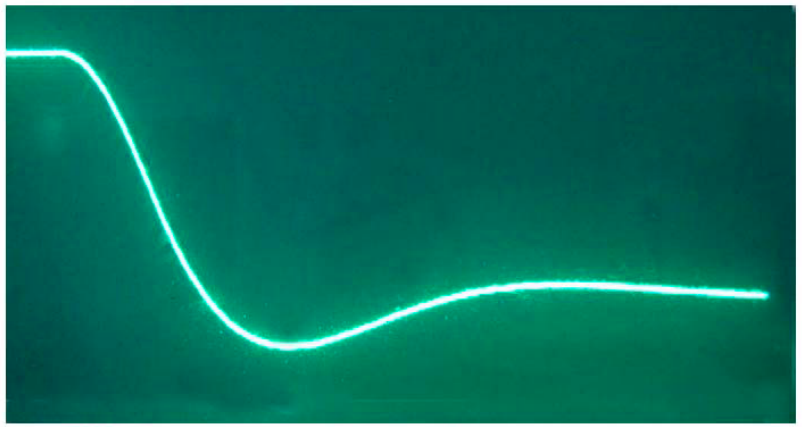


Setting up the computer

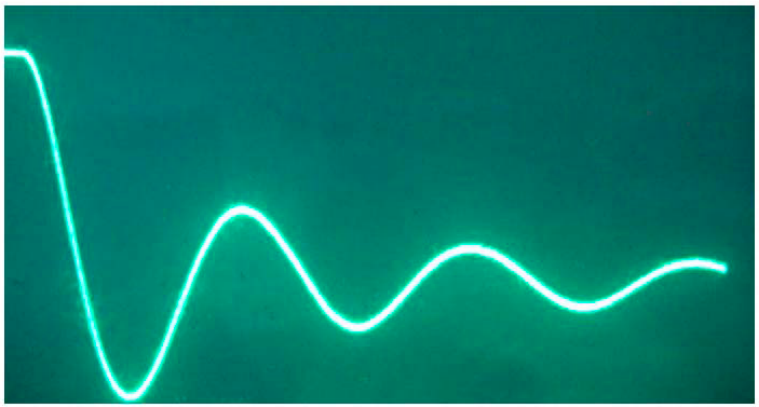


Simulation run with $s = 0.2$ and $d = 0.8$

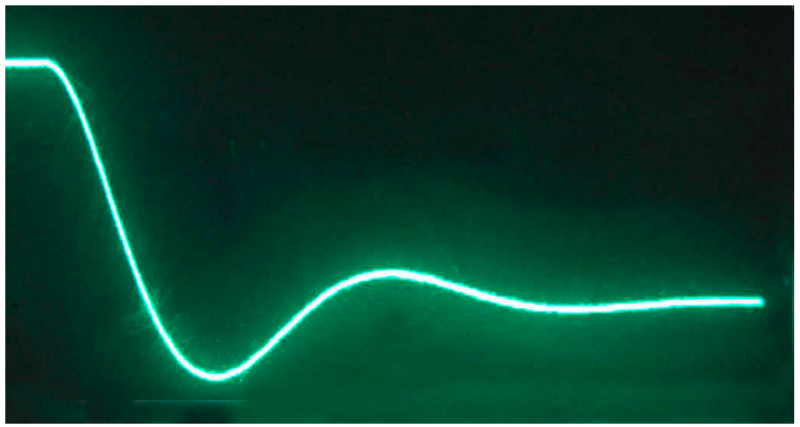
Here and in the following $m = 1$ is assumed.



Simulation run with $s = 0.8$ and $d = 0.6$



Simulation run with $s = 0.4$ and $d = 0.8$



Solving two coupled differential equations

The following example is more complicated than the simple mass-spring-damper system shown before. The goal is to simulate the changes in population numbers in a two species ecosystem populated by rabbits r and lynxes l. Such a system is readily described by Volterra's differential equations:

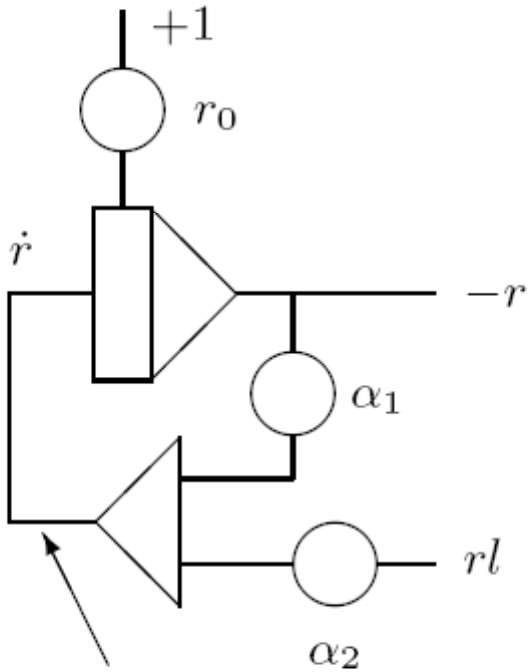
$$\begin{aligned} \dot{r} &= \alpha_1 r - \alpha_2 r l \\ \dot{l} &= -\beta_1 l + \beta_2 r l \end{aligned}$$

The parameters are as follows:

α_1	Rabbit birth rate
α_2	Rate of Rabbits killed by lynxes
β_1	Lynx mortality rate
β_2	Lynx population growth due to killed rabbits

Partial circuit for $\dot{r} = \alpha_1 r - \alpha_2 rl$

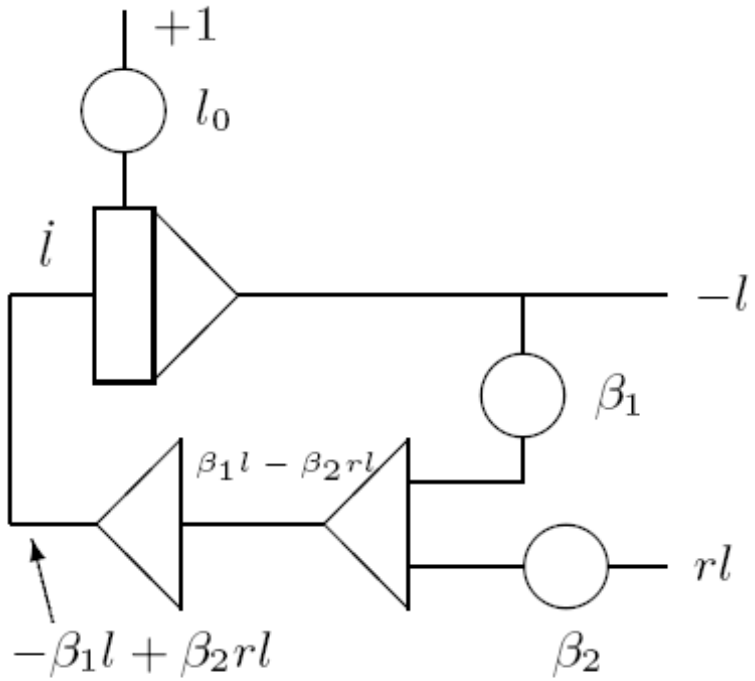
First of all, let us solve $\dot{r} = \alpha_1 r - \alpha_2 rl$ assuming that there is a value rl already known. This leads to the following program:



$$-(-\alpha_1 r + \alpha_2 rl) = \alpha_1 r - \alpha_2 rl$$

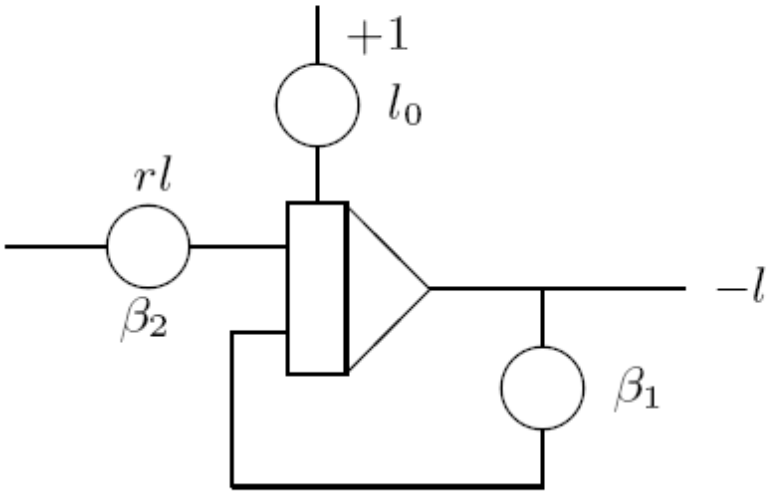
Partial circuit for $\dot{l} = -\beta_1 l + \beta_2 rl$

Next, let us solve $\dot{l} = -\beta_1 l + \beta_2 rl$ – again under the assumption that there already exists a term rl:

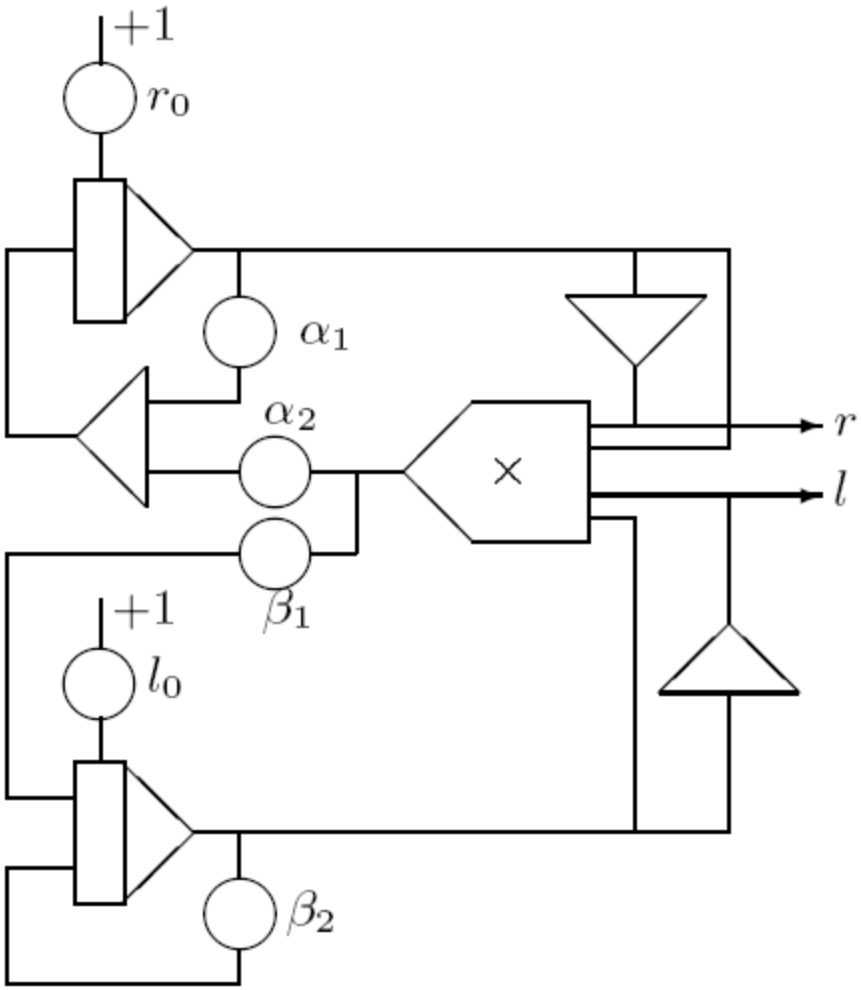


Partial circuit for $\dot{l} = -\beta_1 l + \beta_2 r l$

Obviously we can do better and save two summers:



Coupling both differential equations



$$\dot{r} = \alpha_1 r - \alpha_2 r l$$

$$\dot{l} = -\beta_1 l + \beta_2 r l$$

Scaling the equations

Due to the finite range of values which can be processed by an analog computer, it is necessary to scale the equations to be solved in order to avoid overloading the operational amplifiers and thus introducing erroneous terms.

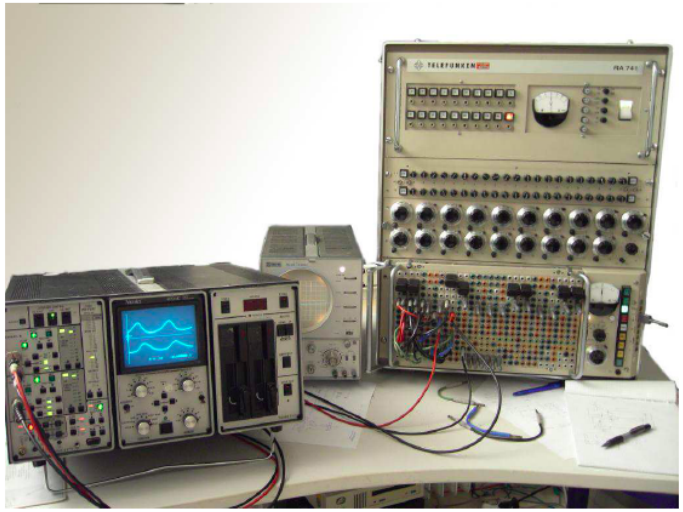
Coupled differential equations like the example above are normally quite difficult to scale since it is challenging to estimate maximum values for the variables.

If a direct scaling is not possible (or if the programmer is too lazy which may be the case much more often) it is possible to run the calculation with a guessed scaling and check for overloads. Then use the values at the moment the overload occurred to determine the next ”guess” and so on.

The values used for the following run were:
$\alpha_1 = 0.17, \alpha_2 = 0.4, \beta_1 = 0.1, \beta_2 = 0.27, r_0 = 0.2, l_0 = 0.8$ (quite unrealistic number of initial lynxes to be honest).

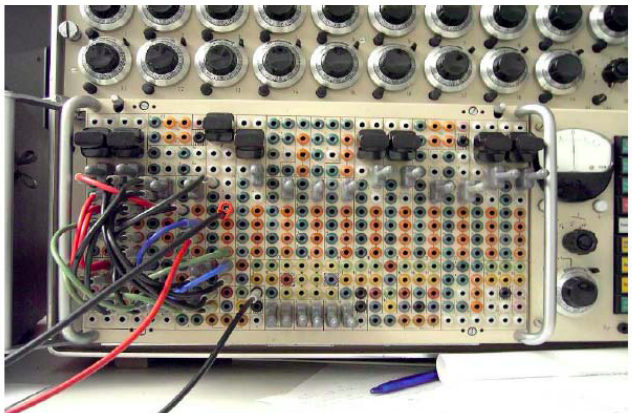
The overall setup

The next picture shows the overall setup featuring a two channel storage oscilloscope:



The completed program

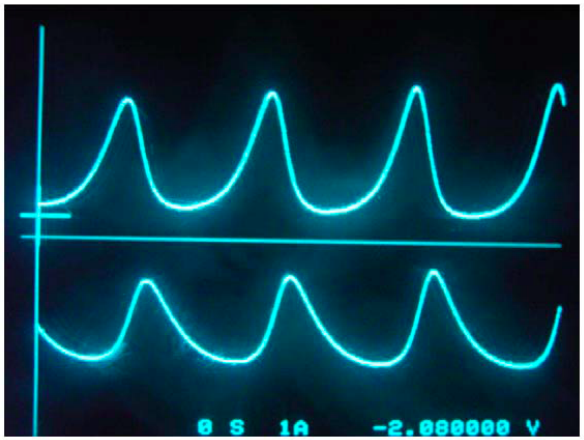
The following picture shows the program as patched for a Telefunken RA741 analog computer:



Running the simulation

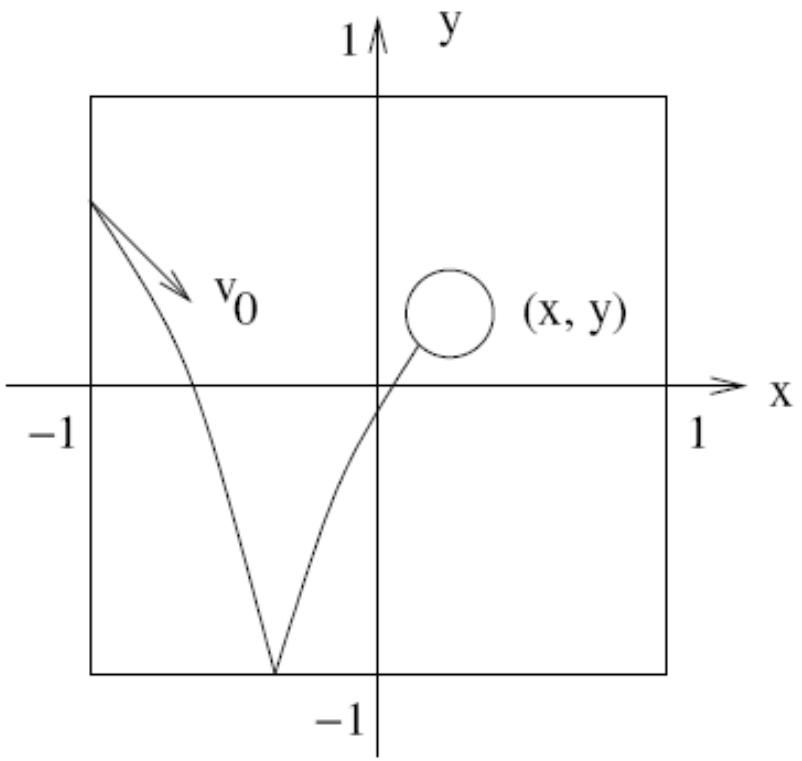
The picture below shows the results of the running simulation:

$$\begin{aligned} \dot{r} &= \alpha_1 r - \alpha_2 r l \\ \dot{l} &= -\beta_1 l + \beta_2 r l \end{aligned}$$



Simulating a ball in a box

The following example is yet a bit more complicated – the simulation of a ball bouncing in a box (cf. [1]) as shown below:



Overall setup of the simulation

The ball is thrown into the left upper corner of the box with an initial velocity of v_0. Whenever it hits a wall of the box it will be reflected elastically.

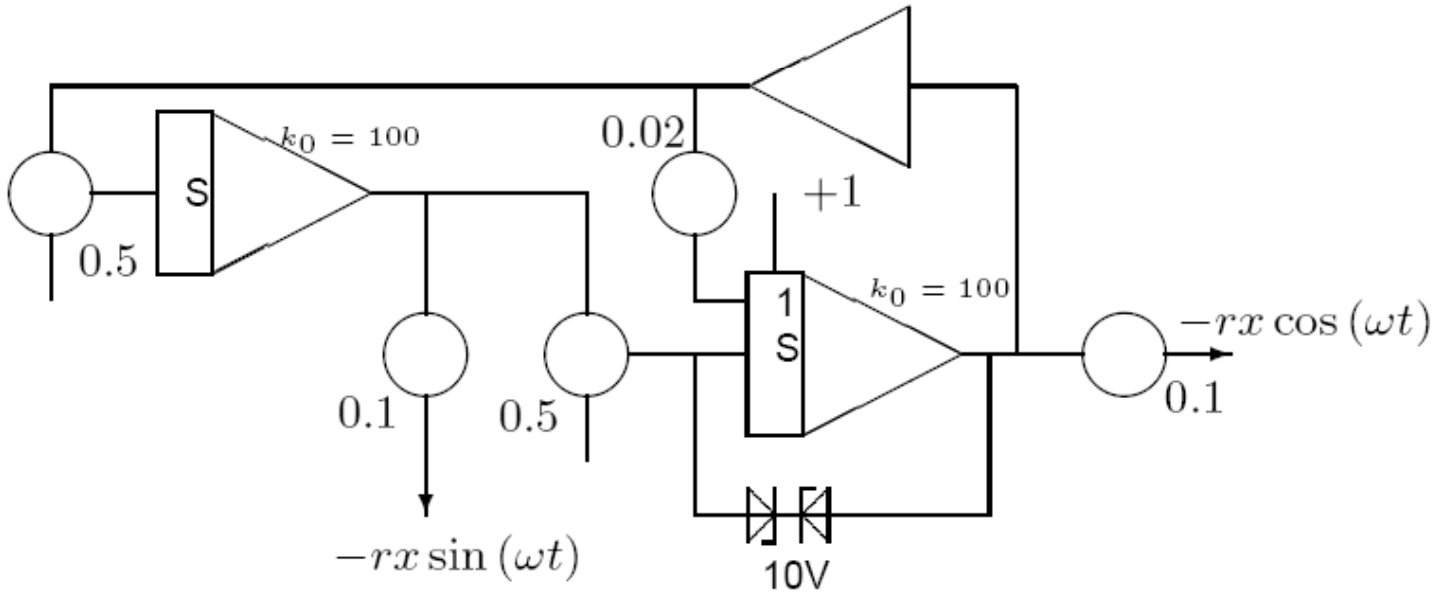
The ball is influenced by a gravitational force pointing downwards and it looses energy by air friction (which is assumed to be proportional to the speed of the ball).

The simulation setup consists of essentially four parts:

1. A $(\sin(\omega t), \cos(\omega t))$-generator to create a real ball instead of a single moving point,
2. a circuit to generate the y-component of the ball's movement in the box,
3. a circuit to generate the x-component and, finally,
4. a summing circuit to overlay these signals in a proper manner.

Generating the ball itself

This is the easiest part of the simulation. A simple sine/cosine-generator with a rather high output frequency is necessary to create the impression of a ball (circle). These two values are generated by solving the well known differential equation $\ddot{y} = -\alpha y$ as shown below:



Generating the ball itself

At the heart of this circuit is the simple sine/cosine-generator constisting of two integrators and a summer.

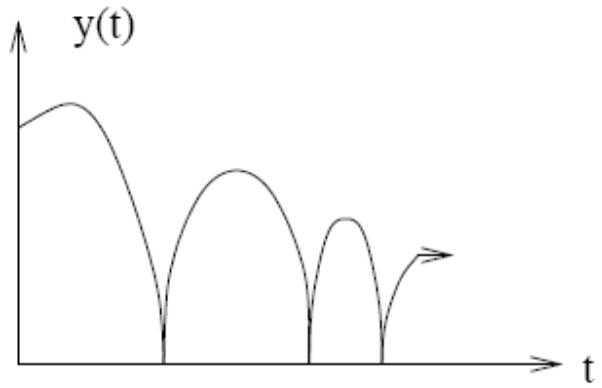
The first thing to note is that the summing junctions of the integrators are used as the main inputs, thus allowing the use of variable input resistances by means of coefficient potentiometers. This is necessary to obtain the desired high output frequency (large ω).

The feedback path from the summer output to the 1-input of the rightmost integrator is used to ”heat up” the oscillation avoiding excessive decay.

The two Zener-diodes are used to avoid overloading the integrator. They will clip the output signal once it reaches one machine unit. This, indeed, will result in a distorted output signal but this distortion is negligible for this application.

Calculating the y-position

The next step towards a complete simulation is the calculation of the y-position of the bouncing ball. Drawing $y(t)$ with t as the free variable results in a graph as shown below:



Calculating $y(t)$

Three terms constitute $\ddot{y}$: The (constant) gravitation, the damping proportional to $\dot{y}$ and the elastic rebound when the ball hits the floor ($y < -1$) or the ceiling of the box:

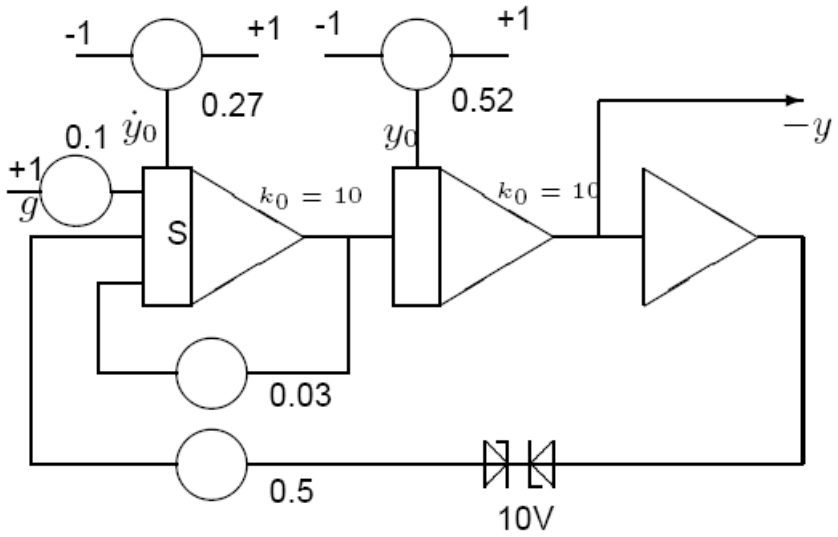
$$\ddot{y} = -g + d\dot{y} \begin{cases} +\frac{c}{m}\,(|y|+1) & \text{if } y < -1 \\ -\frac{c}{m}\,(y-1) & \text{if } y > 1 \end{cases}$$

From $\ddot{y}$ the velocity $\dot{y}$ and position y can be easily derived:

$$\dot{y} = \int_0^T \ddot{y}\,dt + \dot{y}_0$$

$$y = \int_0^T \dot{y}\,dt + y_0$$

Computer setup to calculate $y(t)$



Tricks

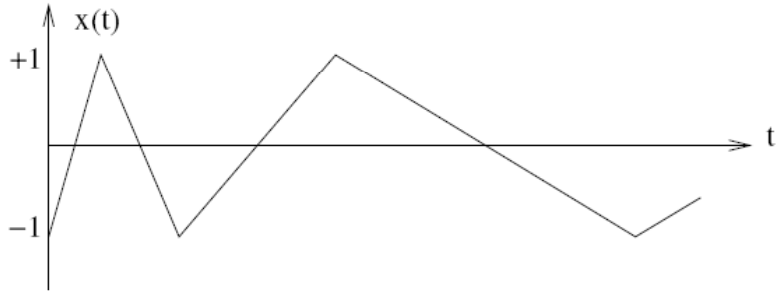
There are some tricks used in this computer setup:

1. The condition of hitting the floor or the ceiling of the surrounding box is detected by the two 10V-Zener-diodes instead of a classical backlash setup. This has the disadvantage that box heights different from ± 1 are not covered as would be possible by using a backlash. The advantage is that two backlashes would require two amplifiers, two potentiometers and four diodes which are saved this way.

2. The slower the ball gets, the smaller the acceleration of the elastic rebounds will be. This is a bit unrealistic and will be partly compensated for by using the summing junction of the first integrator as the input from the simplified backlash instead of using a weighted input.

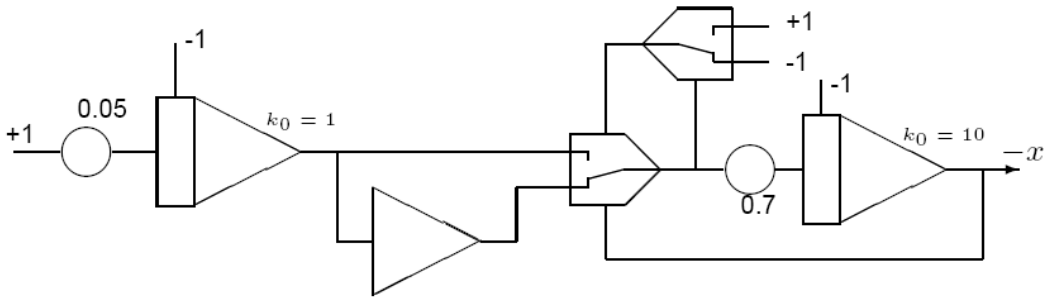
Calculating the x-position

The calculation of $x(t)$ assumes that the velocity diminishes with time t, eventually reaching zero (at this point the computer should enter the halt or initial condition mode).

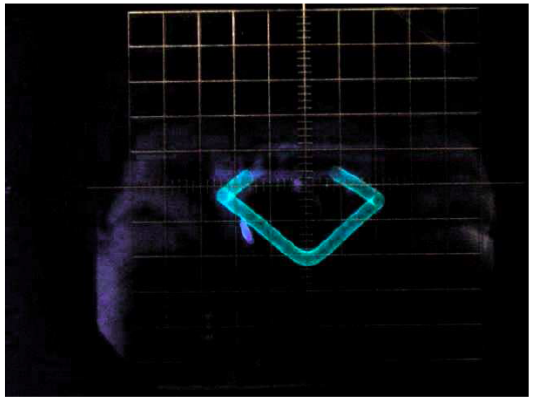


Changing the direction of the ball when it hits the left or right wall is a bit tricky as the following computer setup will show.

Computer setup to calculate $x(t)$

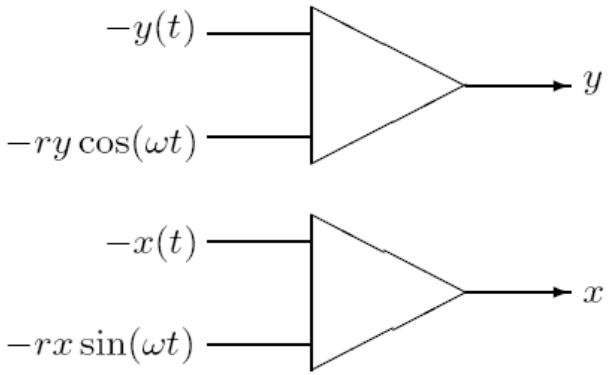


Bouncing ball

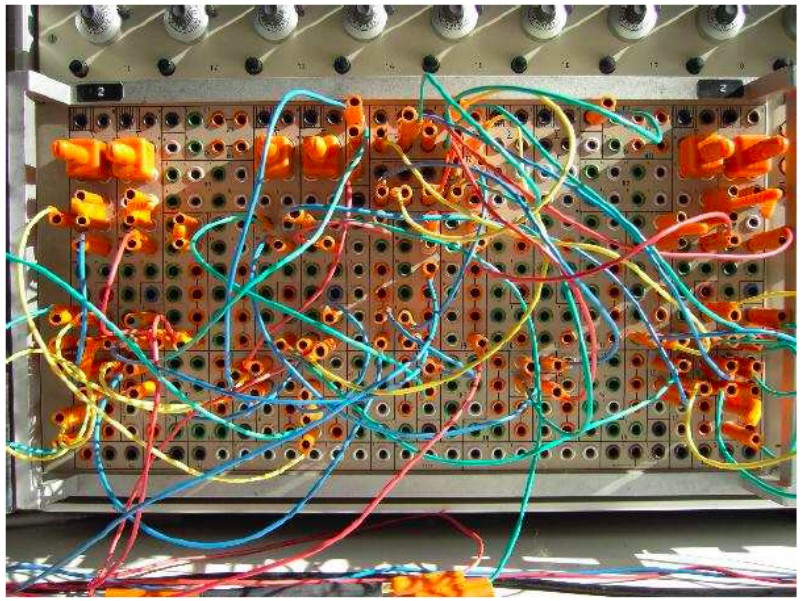


Putting everything together

Now having calculated $y(t)$ as well as $x(t)$ all that is left to do is to superimpose those values with the $(\sin(\omega t), \cos(\omega t))$-pair generated previously to display a real ball at a particular position vector $(x(t), y(t))$:



The final computer setup



Properties of an analog computer

An (electronic) analog computer

- operates on continuous values.
- has a range of values limited by the so called *machine unit* which is normally 10 V or 100 V.
- is inherently parallel and tops (nearly) every digital computer in this respect (apart from DDAs).
- has a quite limited precision.
- is a perfect match for solving scientific and engineering problems which can be described by differential equations.
- is just wonderful!

Computing Faster without CPUs

Scientific Applications on Reconfigurable, FPGA-based* Hypercomputers

by

Dr. Olaf Storaasli
Analytical & Computational Methods Branch
NASA Langley Research Center
Hampton Virginia

2003 Military Aerospace Programmable Logic Device
(MAPLD) International Conference
Reagan Center, Washington DC
September 11, 2003

***Field-Programmable Gate Array**

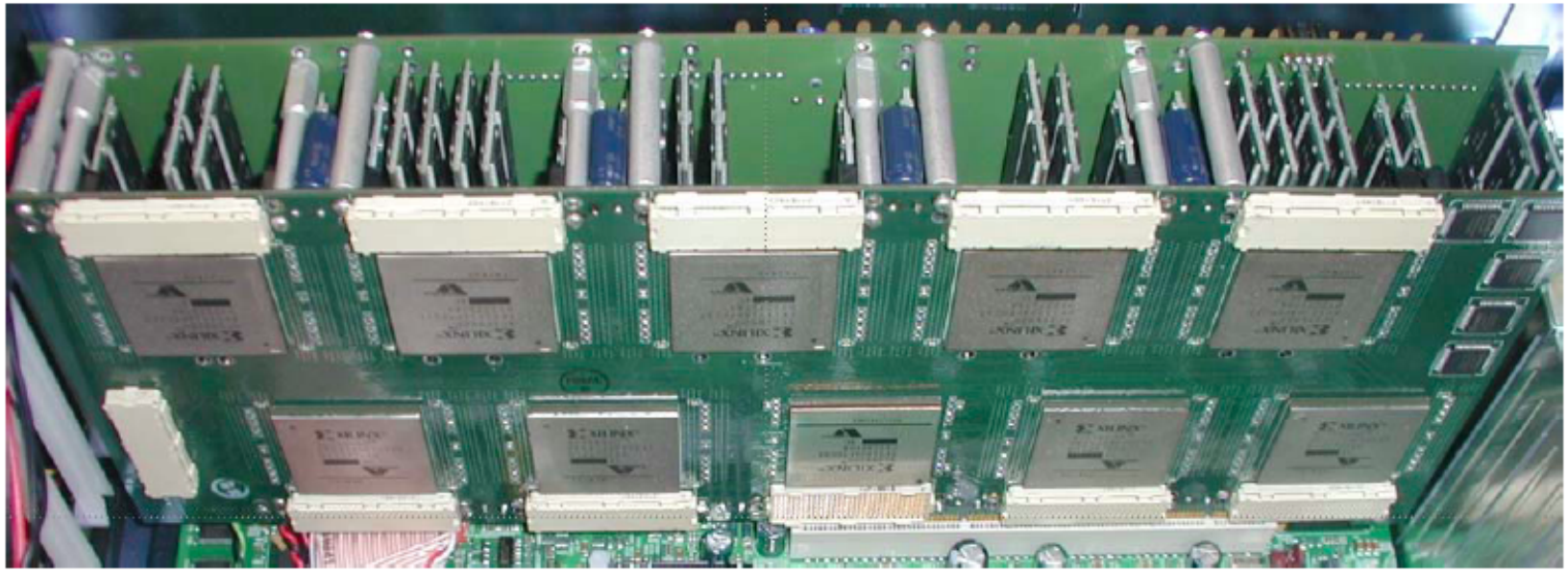
GOAL: Evaluate FPGA-based Hypercomputers for NASA Scientific Computations

TEAM: Drs. Olaf Storaasli & Jarek Sobieski, Principal Investigators

Robert Singleterry, Dave Rutishauser, Joe Rehder, Garry Qualls

Shaun Foley-*MIT*, William Fithian-*Harvard*, Siddhartha Krishnamurthy-*VT*
Cris Kania-*VT*, Patrick Butler-*VT*, Hoy Loper-*GS*, Neha Dandawate-*UVA*,
Kristin Barr-*JPMorgan*, Robert Lewis-*Morehouse* , Vincent Vance-*VT*

PARTNERS: Starbridge Systems, NSA, USAF, MSFC, AlphaStar



FPGA Programming

- User controls gates: *application-specific circuitry*
- Exploit Parallelism
- Reconfigure in milliseconds (new application)
- Code options:
  - **1-D Text**, sequential: ***VHDL, C-to-Gate***
    - Parallelism *esoteric*
  - **2-D Graphic,** parallel *drag & drop*: ***Viva***
    - Parallelism *intuitive*
    - Data flow *like analog computer*

NASA Hypercomputers

Viva 2.0

Project Executable Sheet Edit View System Tools Help

MetaLib **Traditional CPU** **Reconfigurable FPGA**

Sequential: 1 operation/cycle **Parallel**: inherent

Fixed gates & data types **Dynamic** gates & data types

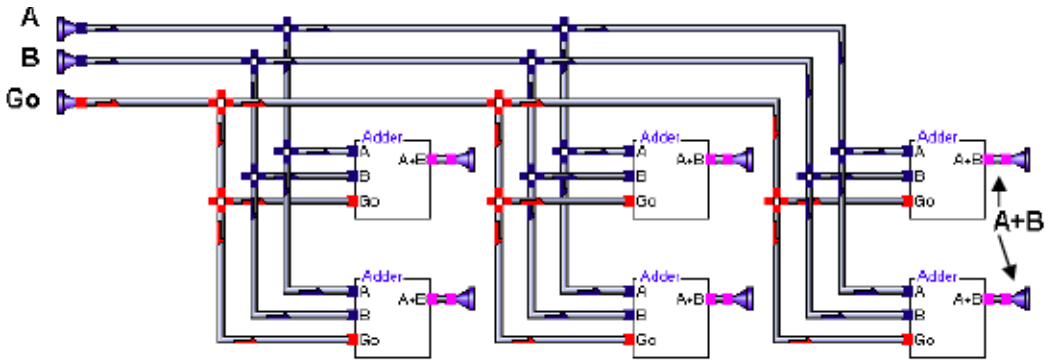
Wasteful: 99% gates idle/cycle **Efficient**: Optimizes gates to task

yet all draw power

Software: Text

```
do i = 1, billion
    c= a+b
end do
```

Gateware: VIVA Icons & Transports



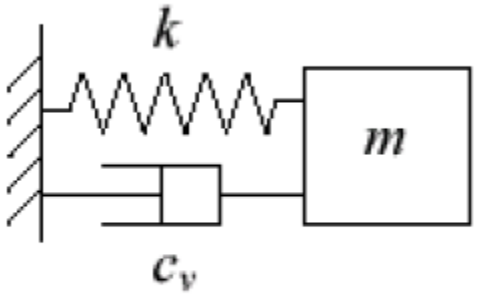
47+ GFLOPS/64 MHz FPGA

26 MFLOPS/250 MHz SGI

329+ GFLOPS/10 FPGA board

FPGA: *New Computing Paradigm*

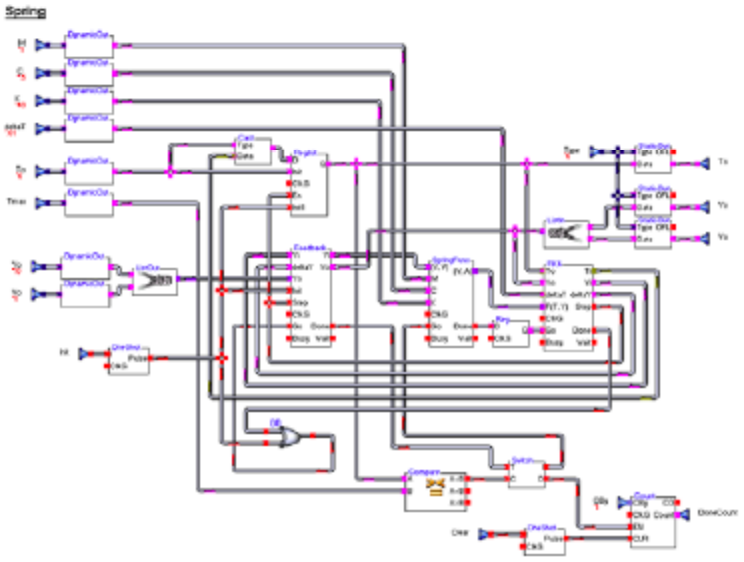
New Project
MetaLib
Basic Data Sets
COM Data Sets
Primitive Objects
Input
Output
$Select
AND
DeRef
INVERT
OR
Ref
Release
Text
Composite Objects
Control
Convert
DataInfo
ExposeCollect
Gates
I/O
Math
Memory
Mux
Registers
Shifting
TestSheets

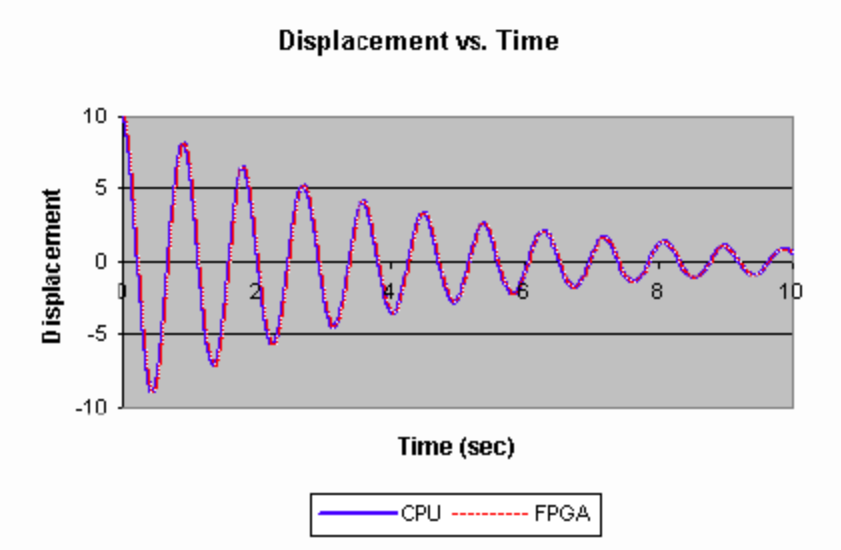


Method: 4-stage Runge-Kutta

$$\frac{du}{dt} = f(u,t) \qquad y(x_0) = y_0$$

$$k_1 = hf(x_n, y_n)$$
$$k_2 = hf(x_n + \tfrac{1}{2}h, y_n + \tfrac{1}{2}k_1)$$
$$k_3 = hf(x_n + \tfrac{1}{2}h, y_n + \tfrac{1}{2}k_2)$$
$$k_4 = hf(x_n + h, y_n + k_3)$$
$$x_{n+1} = x_n + h$$
$$y_{n+1} = y_n + \tfrac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$



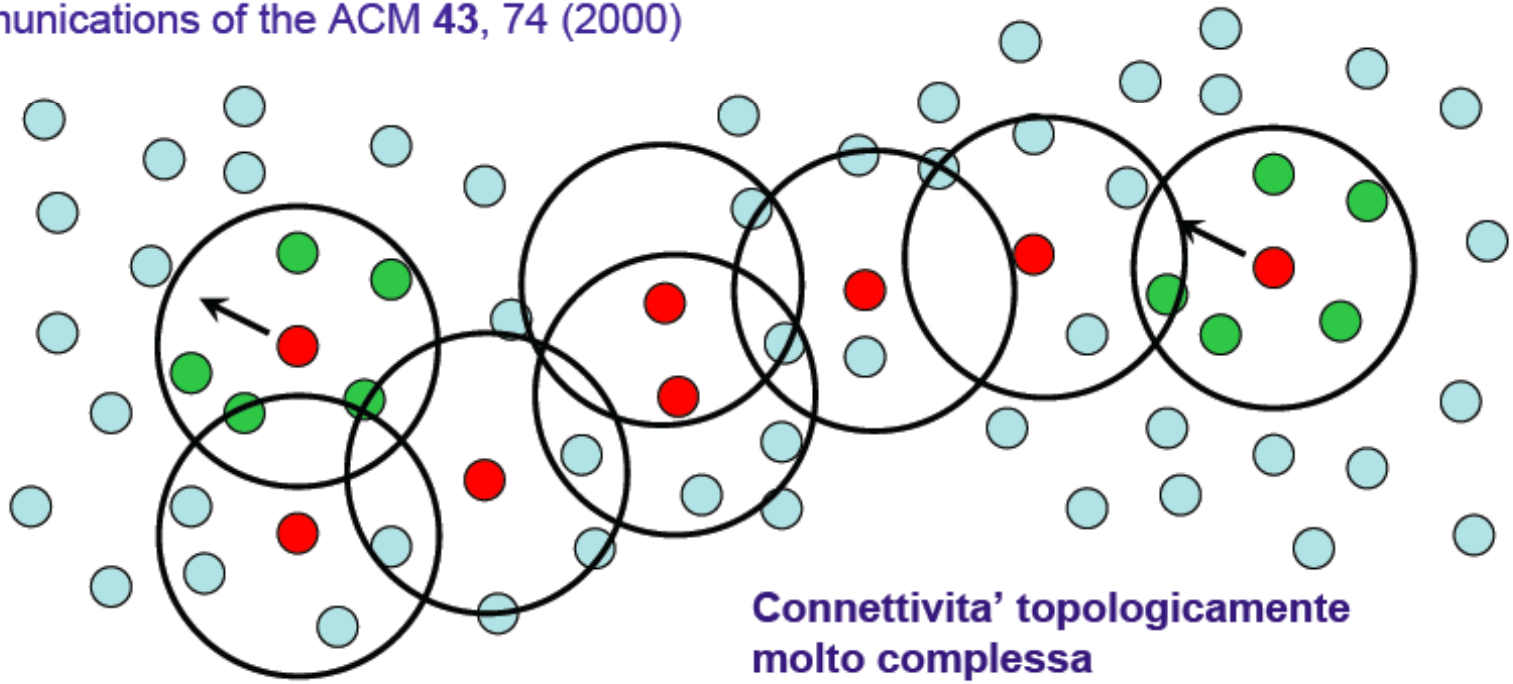


Spring-Mass Solver

Computing amorfo

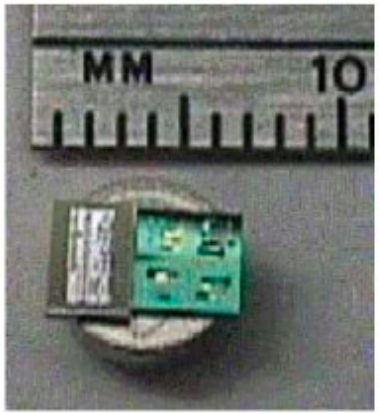
"How does one **engineer** pre-specified, coherent behavior from the **cooperation** of immense numbers of unreliable parts that are interconnected in unknown, irregular, and time-varying ways?"

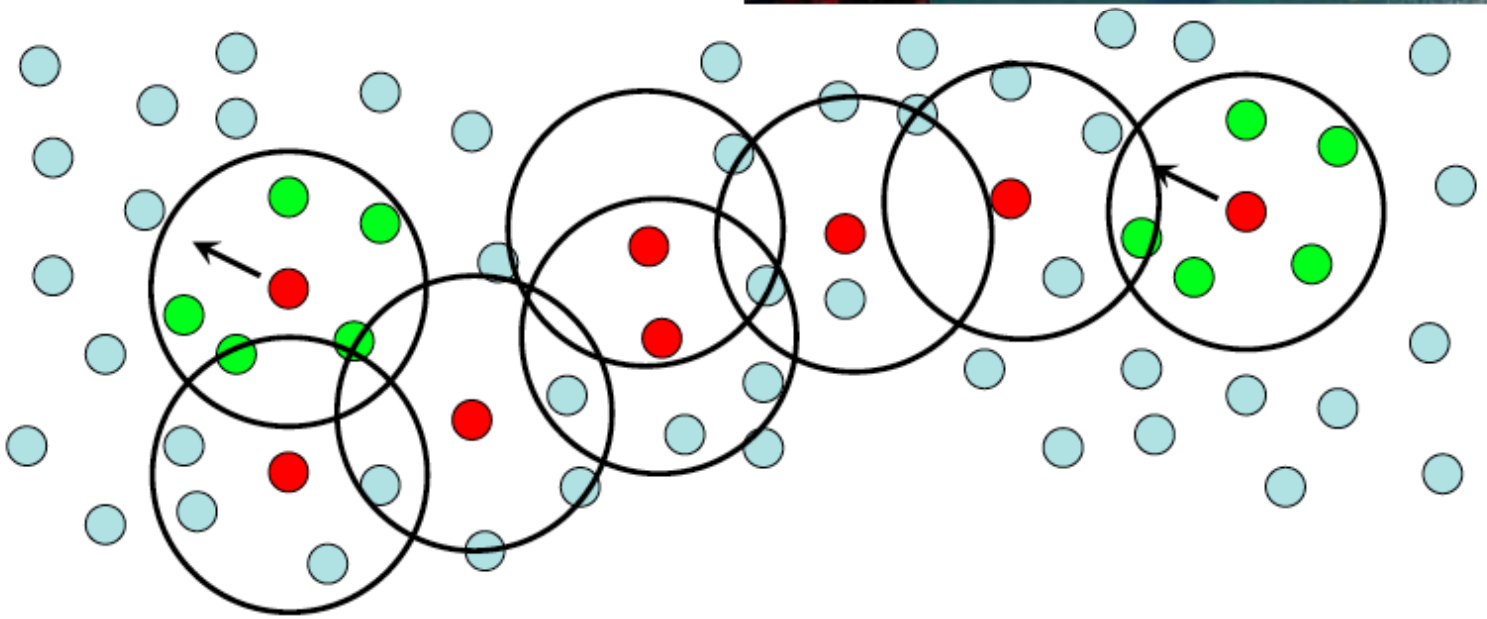
H. Abelson, D. Allen, D. Coore, C. Hanson, G. Homsky, T. Knight, R. Nagpal, E. Rauch, G. J. Sussman, and R. Weiss, *Amorphous Computing*, Communications of the ACM **43**, 74 (2000)



Connettivita' topologicamente molto complessa

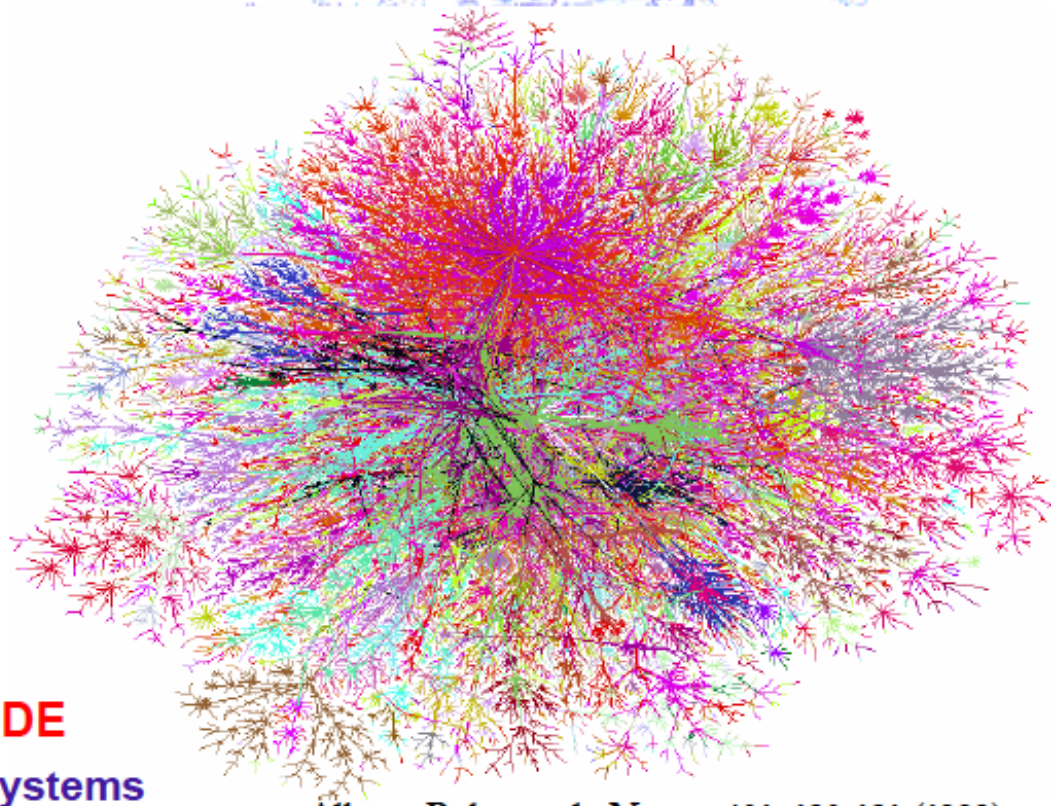
Computing amorfo





Physically-inspired Organizational Paradigms

- **Spatio-temporal Dynamics**
  - Waves, Diffusion
  - Pattern Formation
- **Complexity**
  - Emergent Behavior
  - Self-Organization
  - *Can we tame complexity?*
- **Conservative systems and PDE**
  - Definition of local coordinate systems



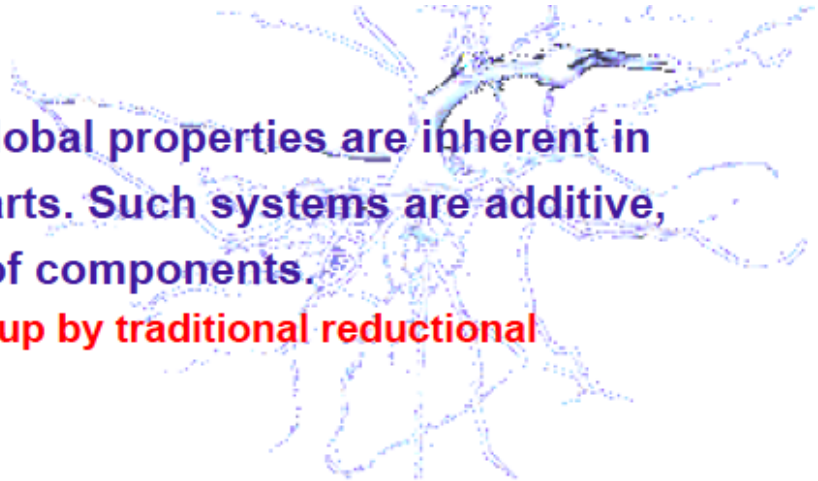
Albert, Reka et al. *Nature* 401, 130-131 (1999).

Paradigmi computazionali e sistemi complessi

- Sistemi con un elevato numero di gradi di liberta': i *processori* elementari sono connessi da una rete di comunicazione topologicamente complessa.

- Comportamento collettivo e naturalmente parallelo poco influenzato dalla natura dettagliata dell'evoluzione temporale del singolo processore e del singolo processo di connessione. Robustezza del funzionamento rispetto a malfunzionamento (distruzione) di elementi funzionali.

- Paradigmi computazionali non definiti a priori in termini algoritmici. La *programmazione* del sistema e' nella sua struttura topologica. La risposta del sistema e' un possibile stato stabile del sistema stesso.



Simple Systems

Simple systems are ones in which global properties are inherent in the properties of their component parts. Such systems are additive, and scale with increasing numbers of components.

Can be studied top-down or bottom up by traditional reductional science.

Complex Systems

They can be defined by what they are not:

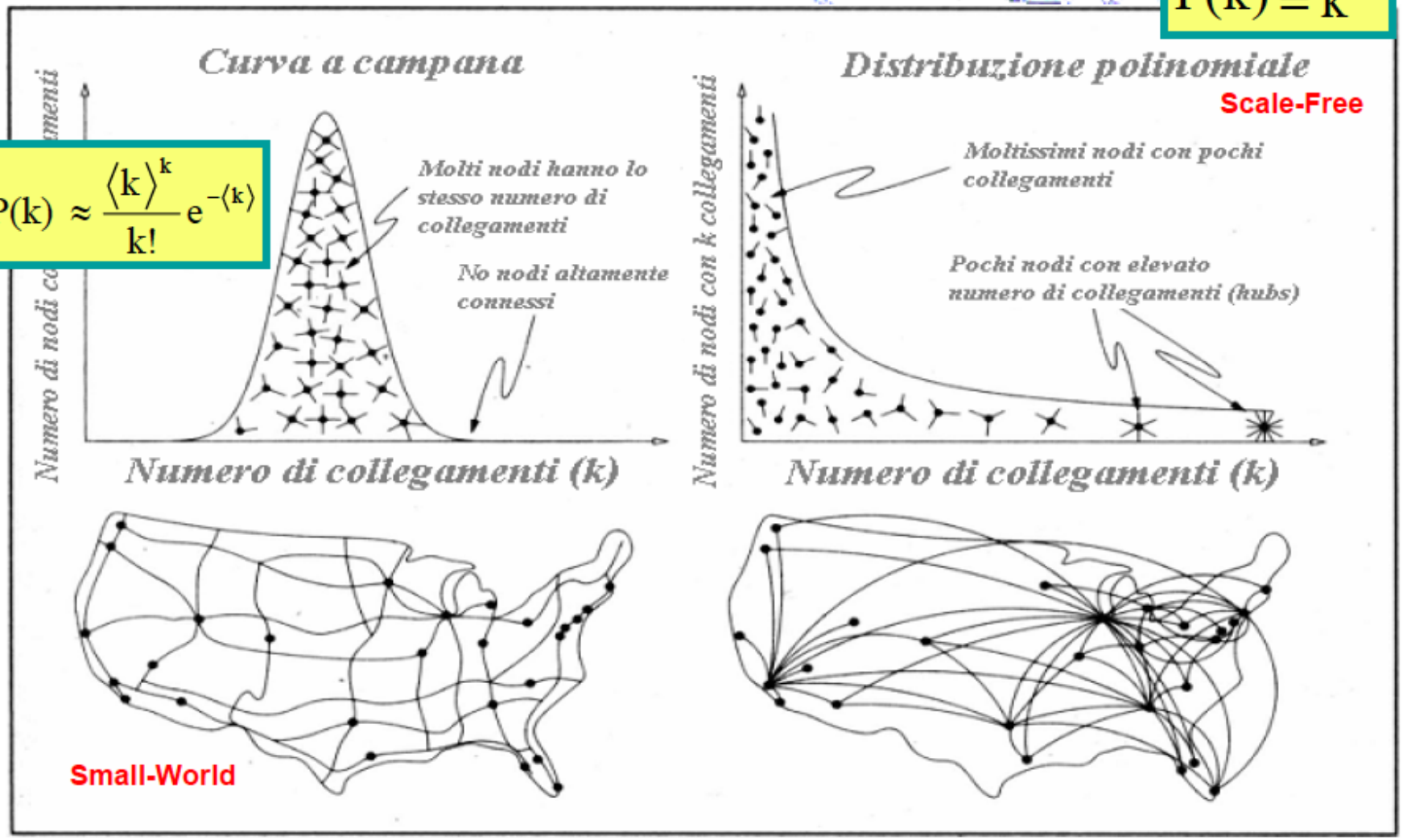
Complex systems are not simple ones.
The fundamental characteristic of a complex system is that it exhibits emergent properties.

Emergent properties are ones that arise due to the interactions in a system, and are not inherent in the individual components

There are almost as many definitions of CxSys as there are CxSys researchers. Many definitions include a notion of “surprise”. Emergent properties can be surprising, but equating emergence with surprise is a statement about the human observer, not the system

Many complex interacting systems can be described by networks (graphs)

$$P(k) \cong k^{-3}$$



$$P(k) \approx \frac{\langle k \rangle^k}{k!} e^{-\langle k \rangle}$$

Topologia & Epidemie

Il modello SIS
(Susceptible-Infected-Susceptible)

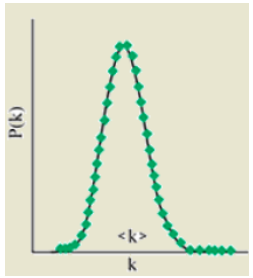
- un nodo sano è infettato con probabilità **υ** se almeno un vicino è infetto
- i nodi infetti sono curati con probabilità **δ** e divengono infettabili

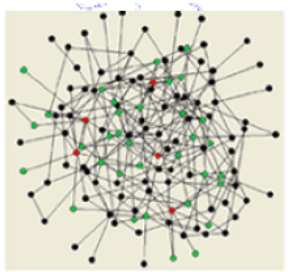
- Modello topologico di sistema complesso

Rate di diffusione $\lambda = \upsilon/\delta$

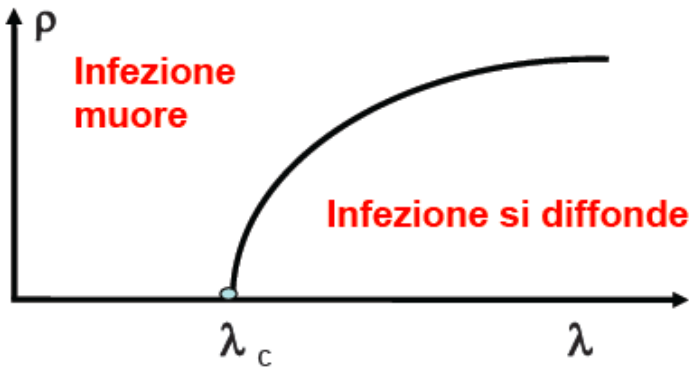
Nodo ⟺ Individuo

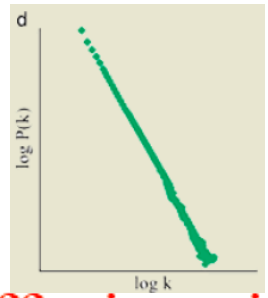
Link ⟺ Connessione lungo cui può propagarsi l'infezione

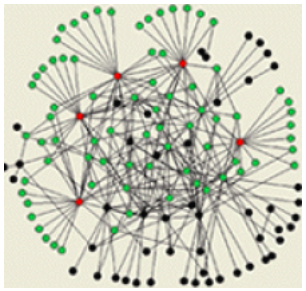




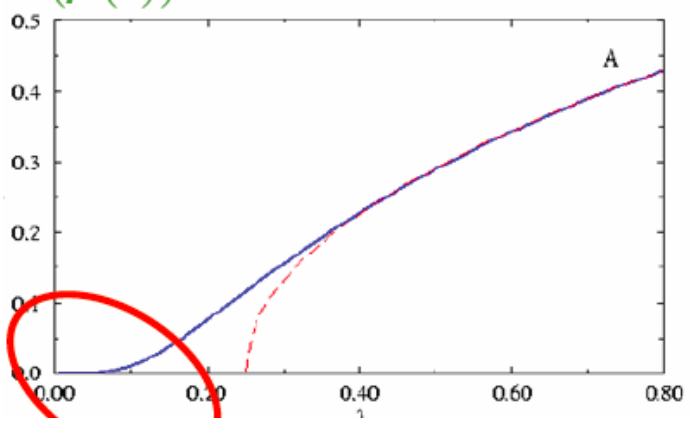
Modello esponenziale

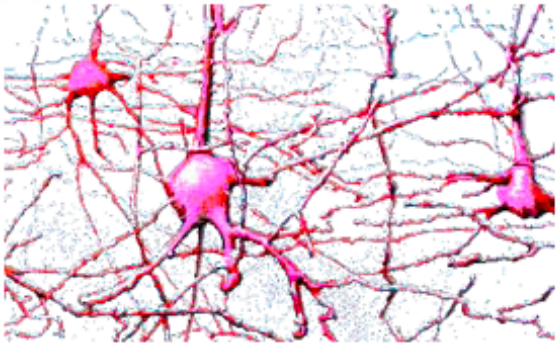






Diffusione in reti polinomiali

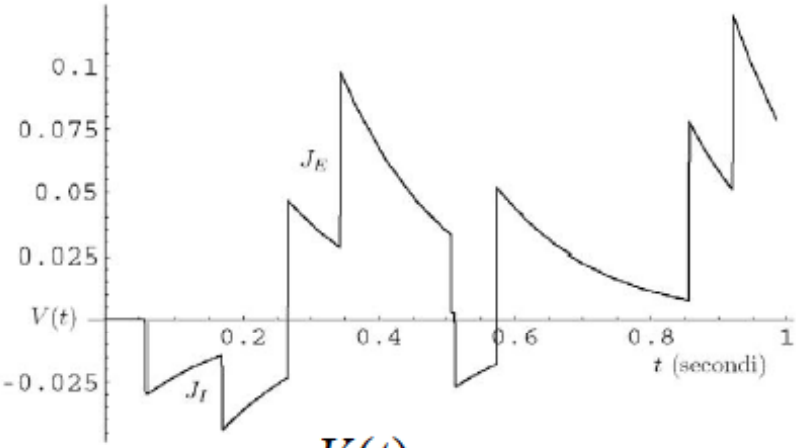




In sostanza:

$$\begin{cases} \dfrac{dV(t)}{dt} = -\dfrac{V(t)}{\tau} + I(t) & \text{se } V(t) < \vartheta \\ V(t) = 0 & \text{se } V(t) > \vartheta \end{cases}$$

- ✓ **Numero di connessioni afferenti sul singolo neuroni ~10^4**
- ✓ **basse frequenze di emissione**
- ✓ **$J_{ij} << \vartheta$**
- ✓ **Probabilità che 2 neuroni siano direttamente connessi ~10%**



$$dV(t) = \frac{-V(t)}{\tau}dt + J_e dN_e(t) + J_i dN_i(t)$$

$$dN(t) = N(t+\Delta t) - N(t)$$

$$\Pr\{dN(t) = n\} = \frac{[Cv(t)dt]^n e^{-Cv(t)dt}}{n!}$$

$$v(t) \equiv \frac{1}{C}\sum_{j=1}^{C} v_j\left(t - d_j\right)$$

Processi di emissione indipendenti. *V(t)* è un processo stocastico a salti discreti guidato da un rumore Poissoniano

C = numero di connessioni

***v(t)* = correnti di probabilita'**

Nel limite $C_\alpha \to \infty$ e $J_\alpha \to 0$ $\alpha = I, E$

Processo stocastico discreto di Poisson

Processo di depolarizzazione

$$dV(t) = \frac{-V(t)}{\tau}dt + \mu dt + \sigma d\Gamma(t)$$

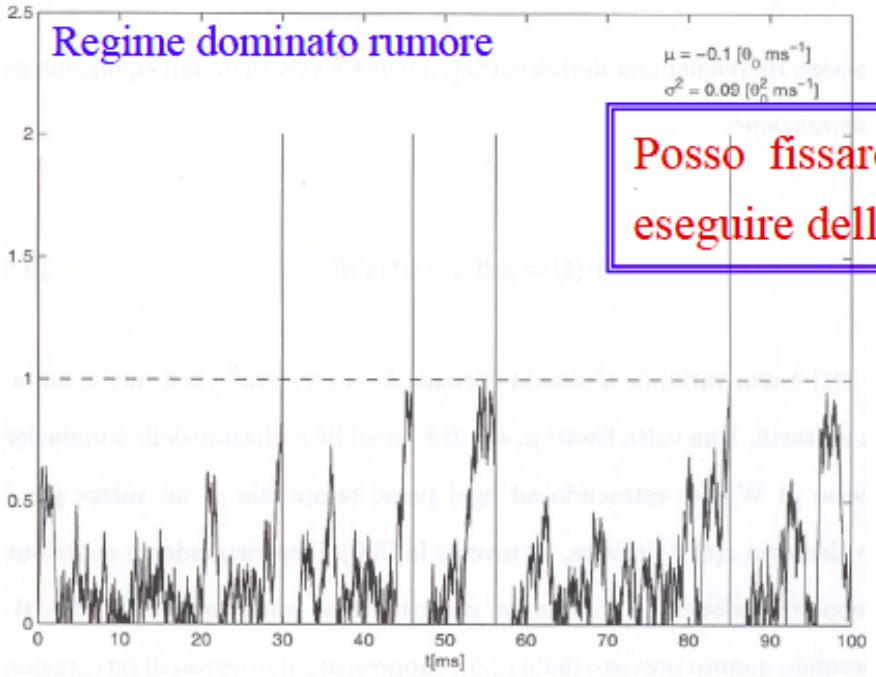
$$\langle d\Gamma(t) \rangle = 0 \quad \langle d\Gamma(t)^2 \rangle = \mathrm{dt}$$

con $\mu dt = \langle dV \rangle$ e $\sigma^2 dt = Var[dV]$

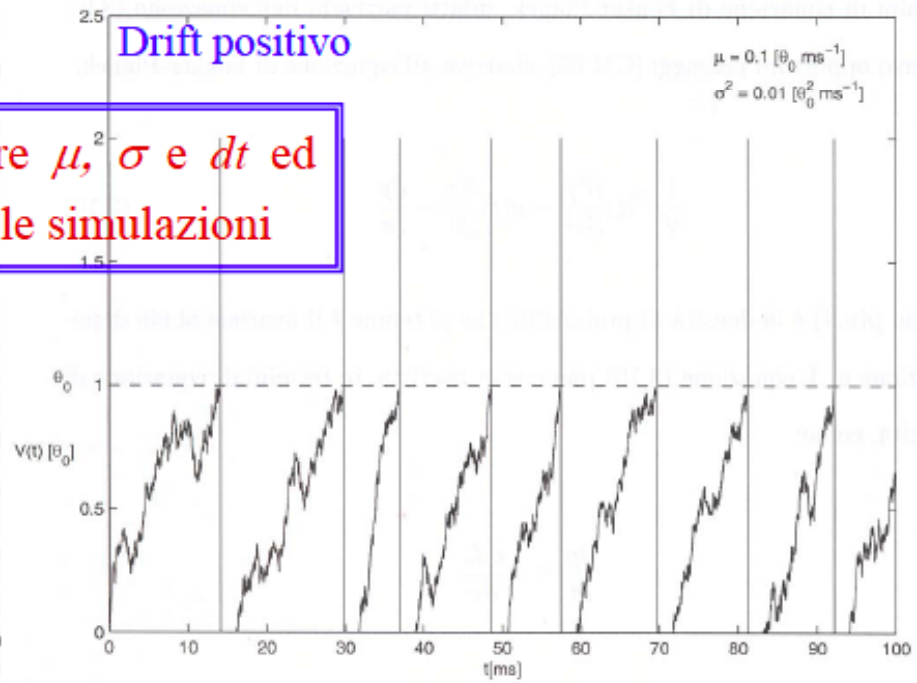
Processo stocastico continuo di Wiener

$$C_E = \frac{\sigma^2 / J_I + \mu}{J_E(1 + J_E/J_I)\nu_E}$$

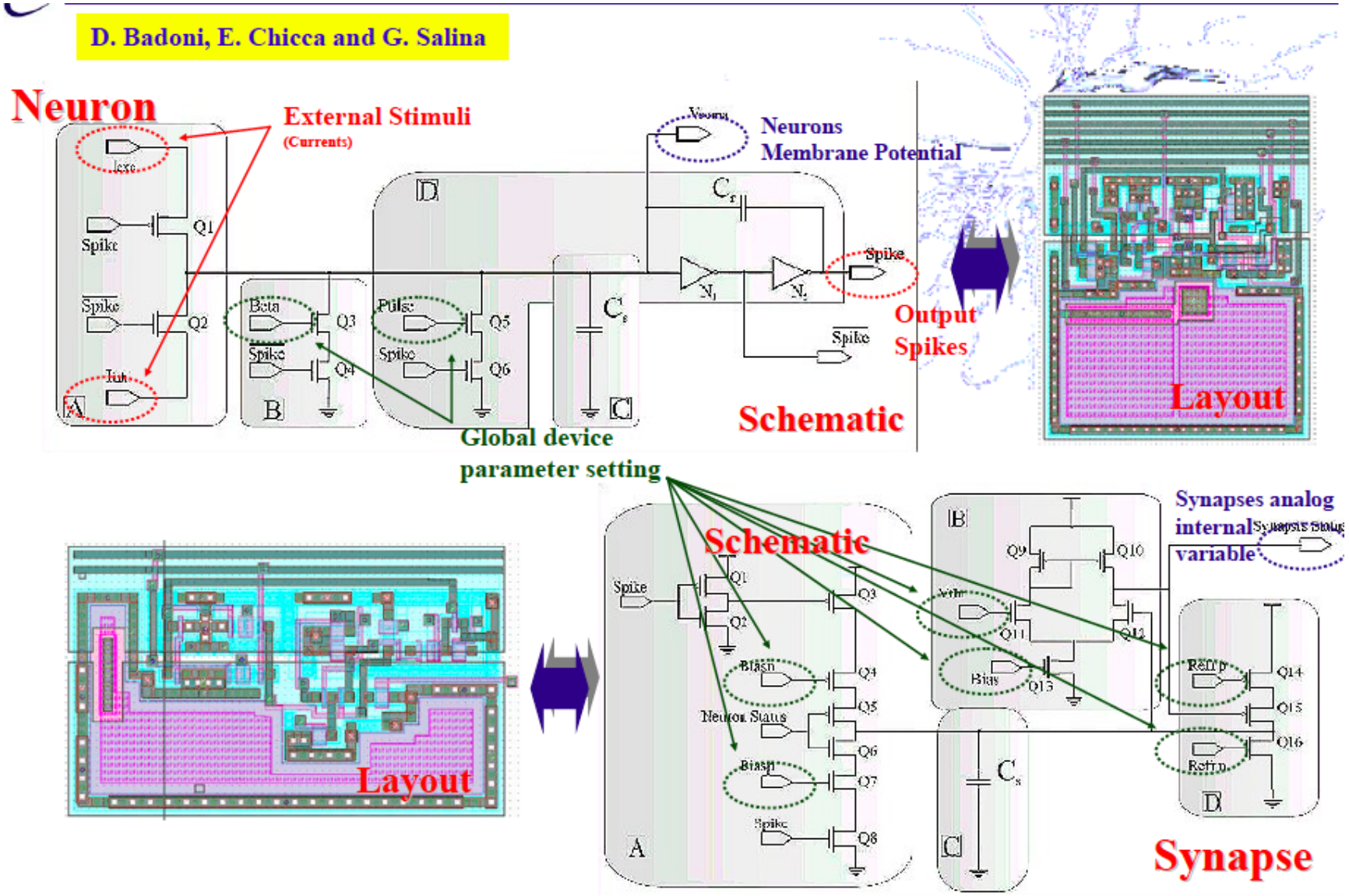
$$C_I = \frac{\sigma^2 / J_E - \mu}{J_I(1 + J_I/J_E)\nu_I}$$



Posso fissare μ, σ e dt ed eseguire delle simulazioni

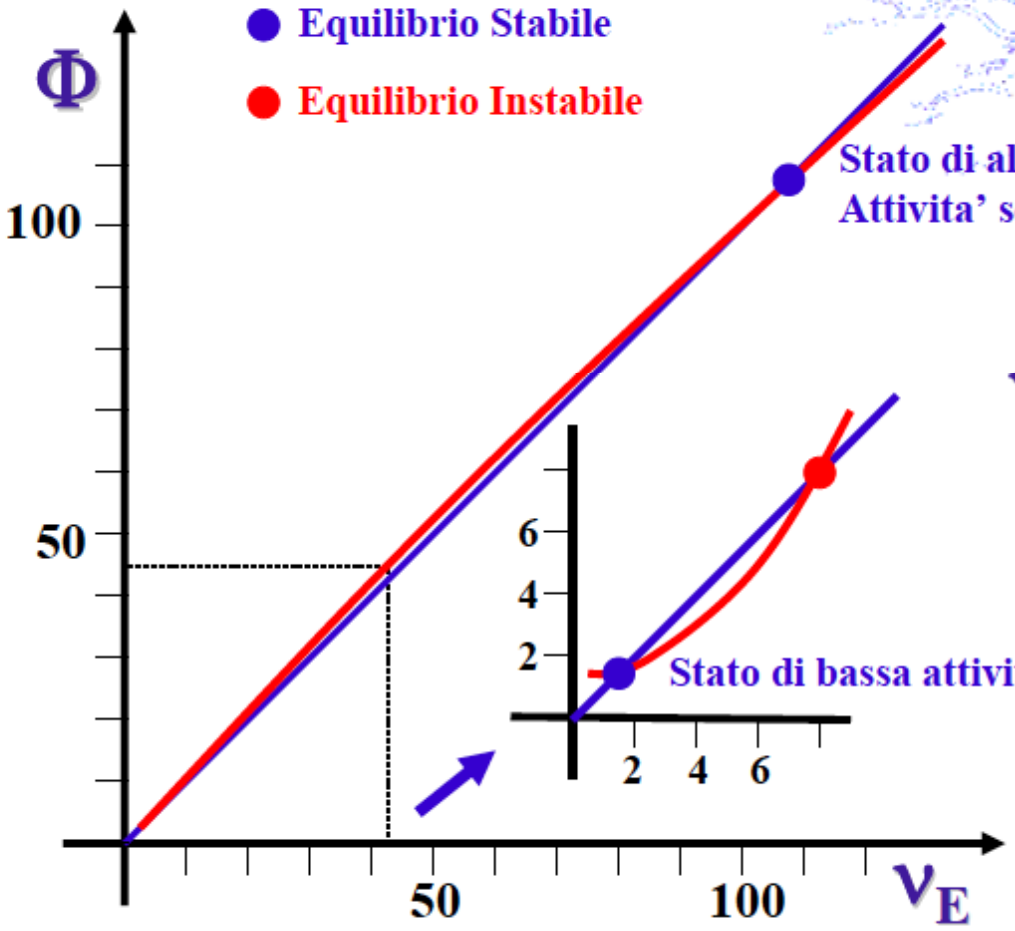


D. Badoni, E. Chicca and G. Salina



Il neurone Integrated and Fire lineare

Equazione di autoconsistenza per le frequenza di emissione



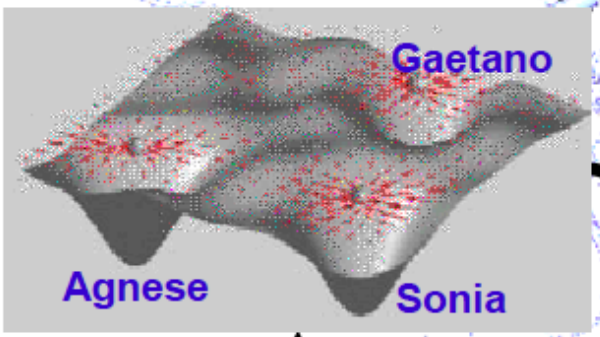
Stato di alta attivita'
Attivita' selettiva

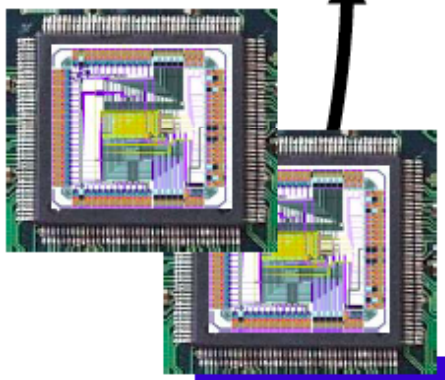
$$\nu_E = \Phi(\mu_E, \sigma_E)$$

Stato di bassa attivita'. Attivita' spontanea

Neuroni Impulsati & Modelli Cognitivi

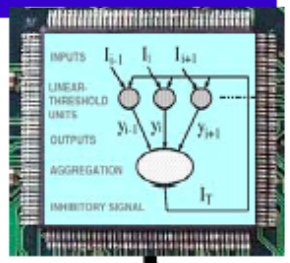
Processi di classificazione in tempo reale biologicamente plausibili.

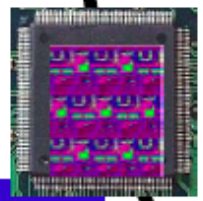




a-VLSI Networks

WTA Network





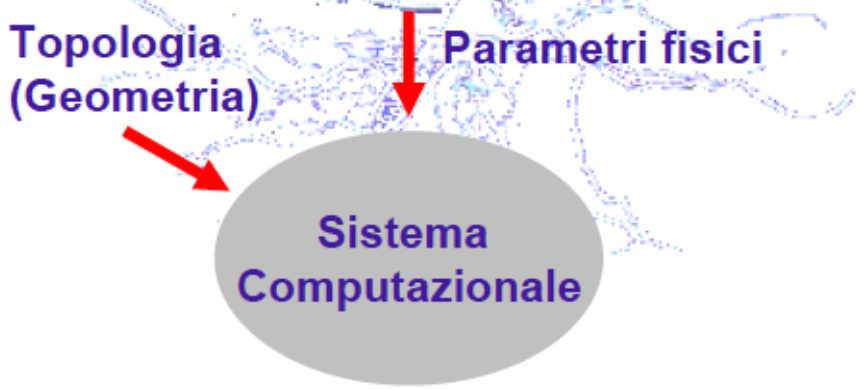
Retina

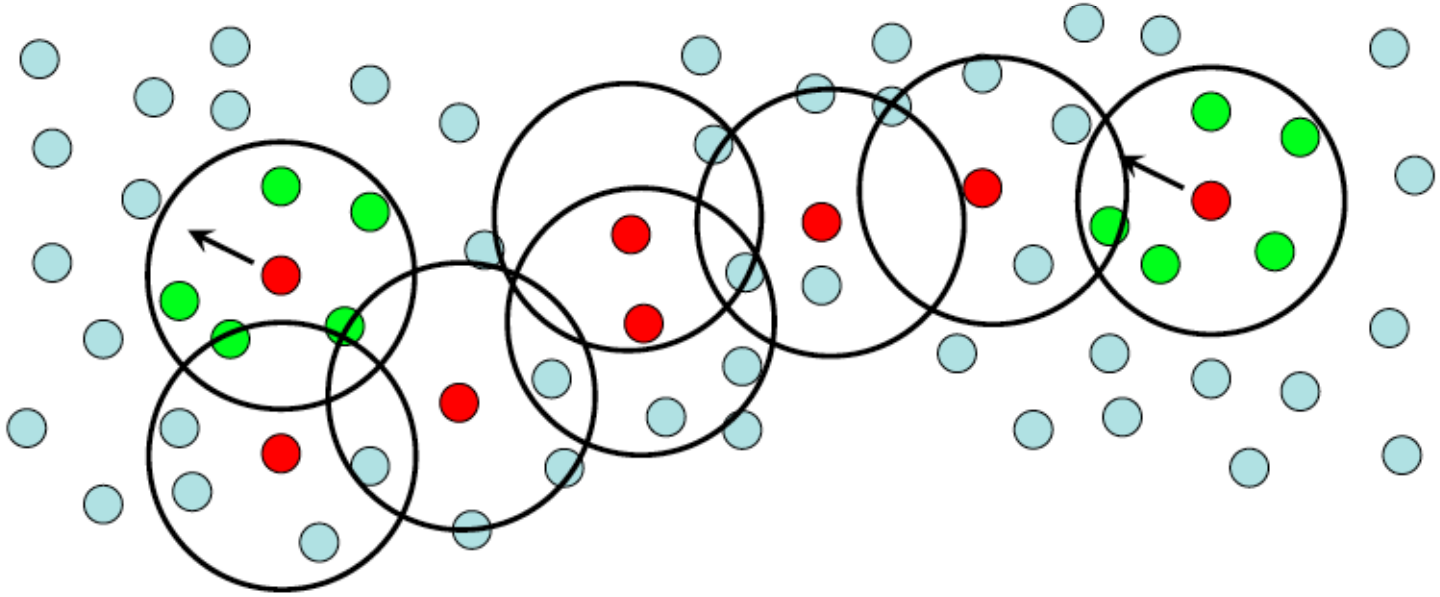
Agnese!

Abbiamo imparato qualcosina ???? !!!!

Complexity

- Emergent Behavior
- Self-Organization
- *Can we tame complexity?*





E vissero tutti felici e contenti !

?