Workshop – Analog Computing

Bernd Ulmann

19-APR-2006

Structure of this workshop

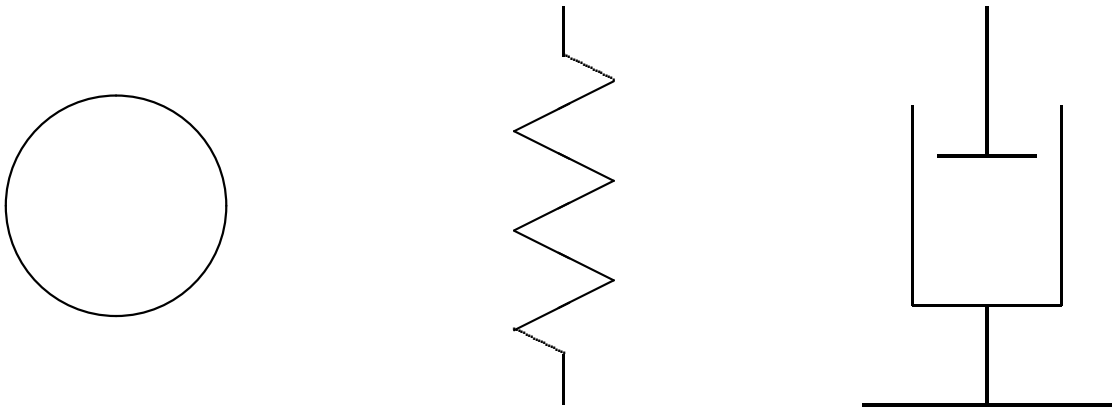
The following slides are part of the workshop ”Analog Computing” which was held at the VCFE-2006 in Munich.

This workshop gives an introduction to the art of analog computing by outlining some examples ranging from a mass-spring-damper system to a bouncing ball in a box.

All examples have been programmed and executed on real analog computers like the Telefunken RA742, etc.

Simulating a mass-spring-damper system

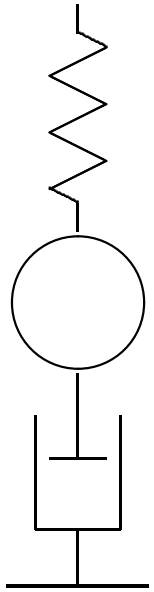
The first example shows how to simulate the behaviour of a rather simple mechanical system consisting of a mass, a spring and a damper. The basic elements of this system are shown below with the mathematical representation of the forces belonging to each:



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

Putting all elements together

Connecting these three elements together yields (thanks to nature and Newton) the following setup and equations:



$$F_m + F_d + F_s = 0$$

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

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:

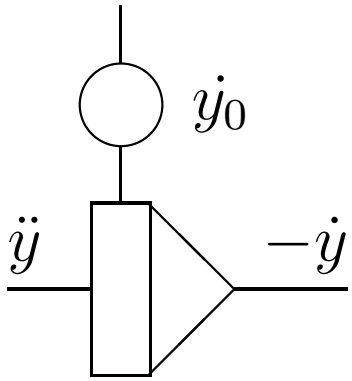
$$\ddot{y} = -\frac{1}{m}\left(d\dot{y} + sy\right).$$

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!

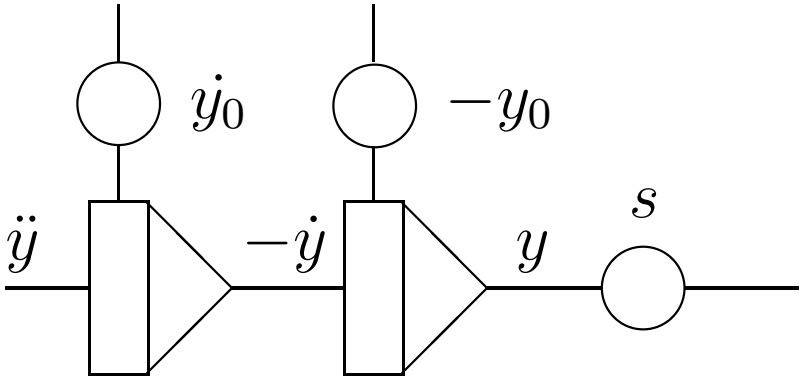
Generating $-\dot{y}$

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:



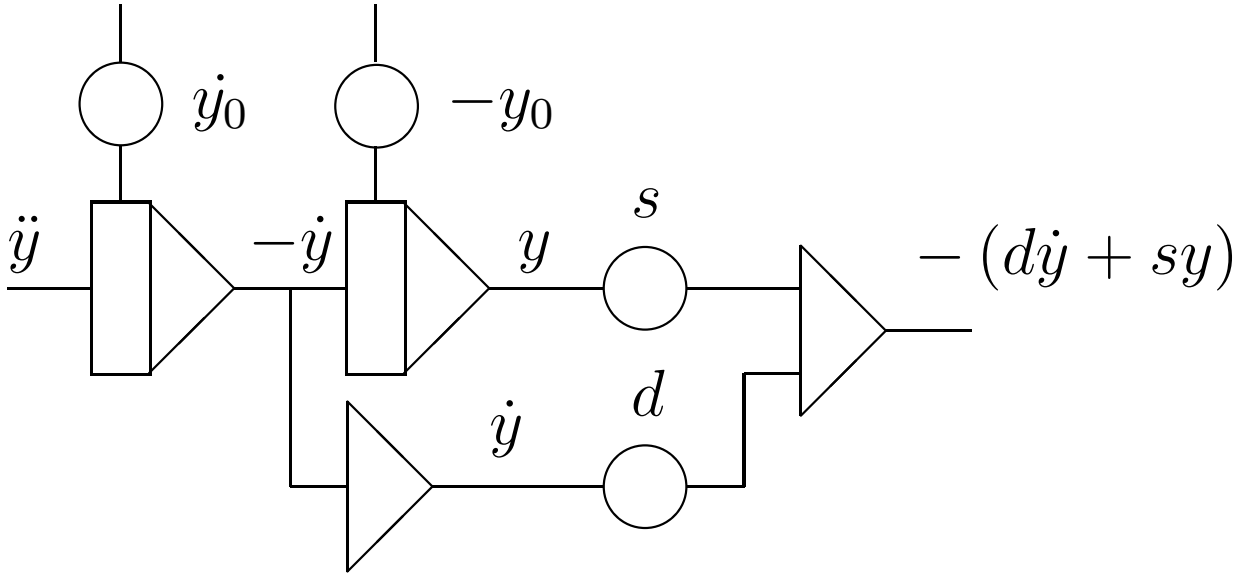
Generating $F_s = sy$

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



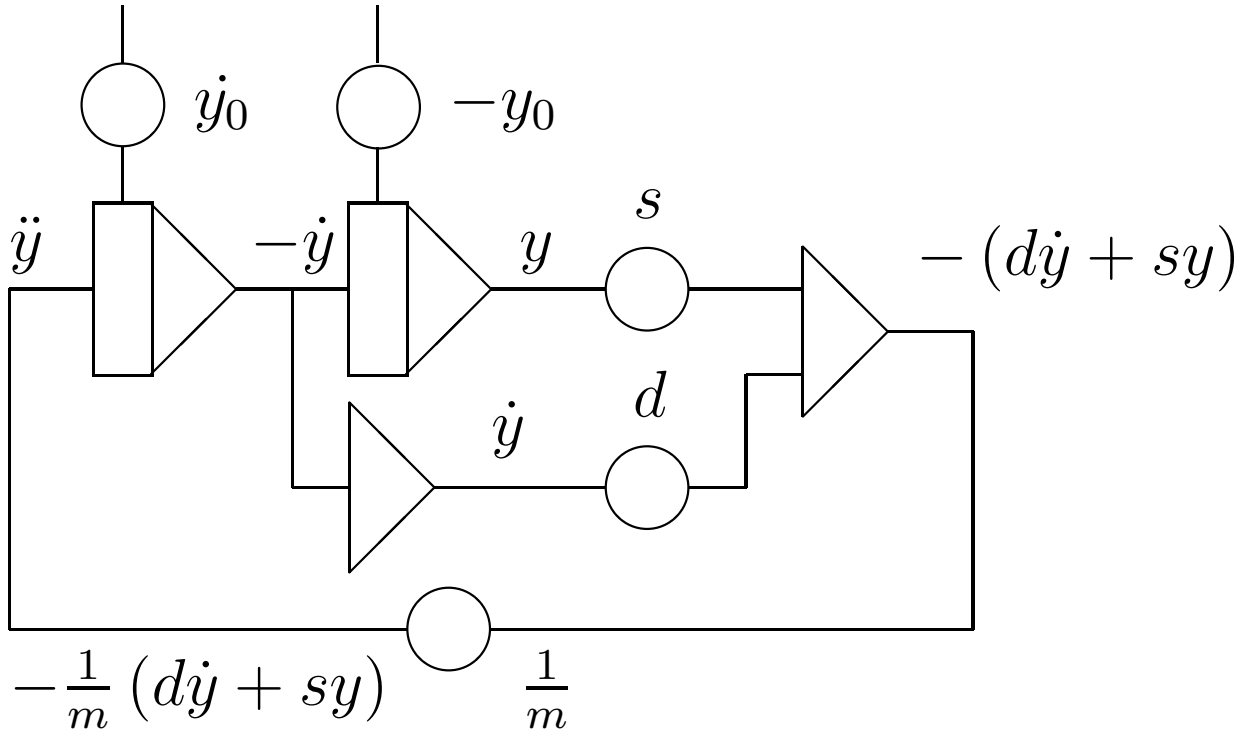
Generating F_d and the sum of forces

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:

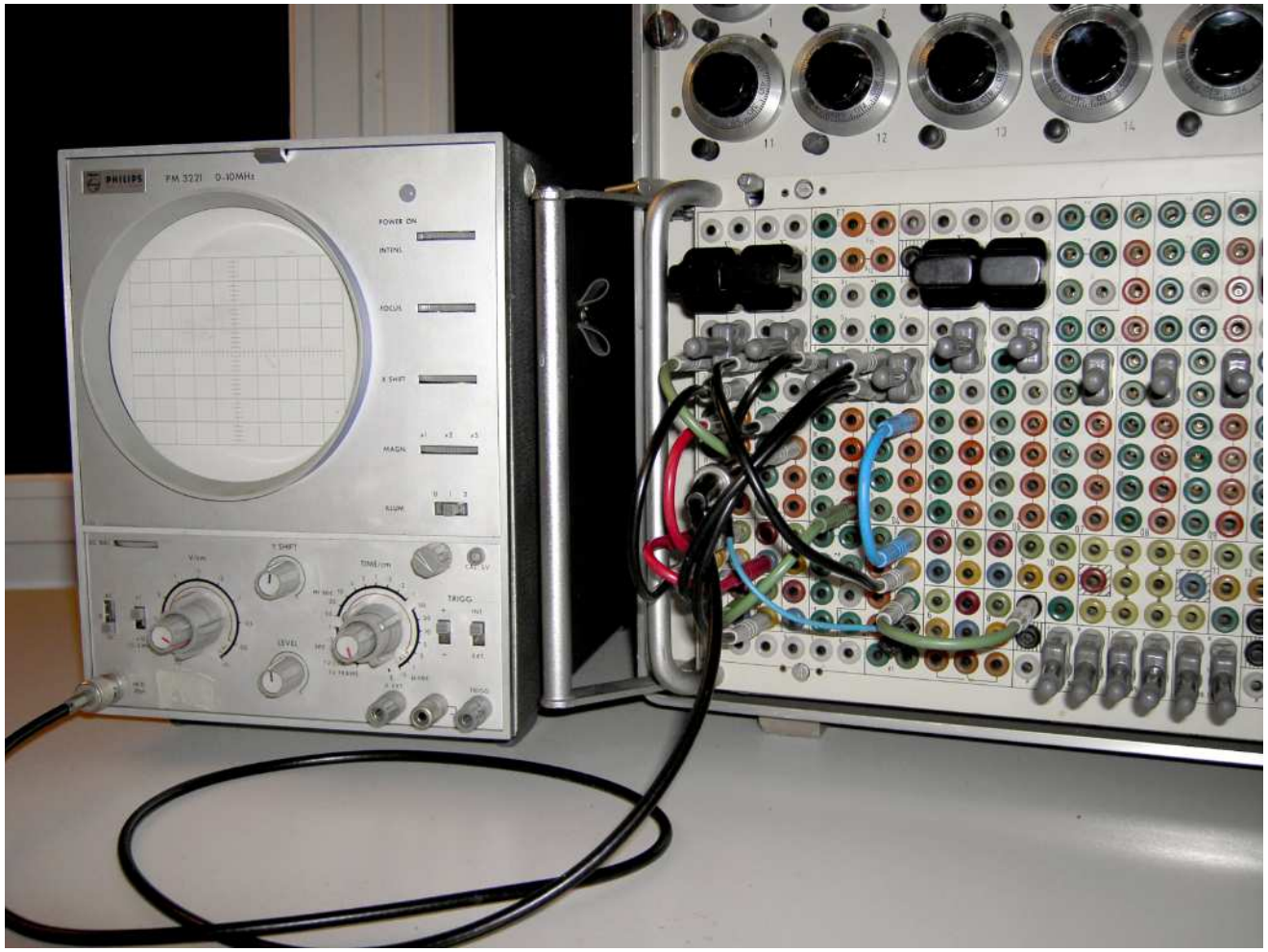


Closing the loop

The sum $-(F_s + F_d)$ 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:

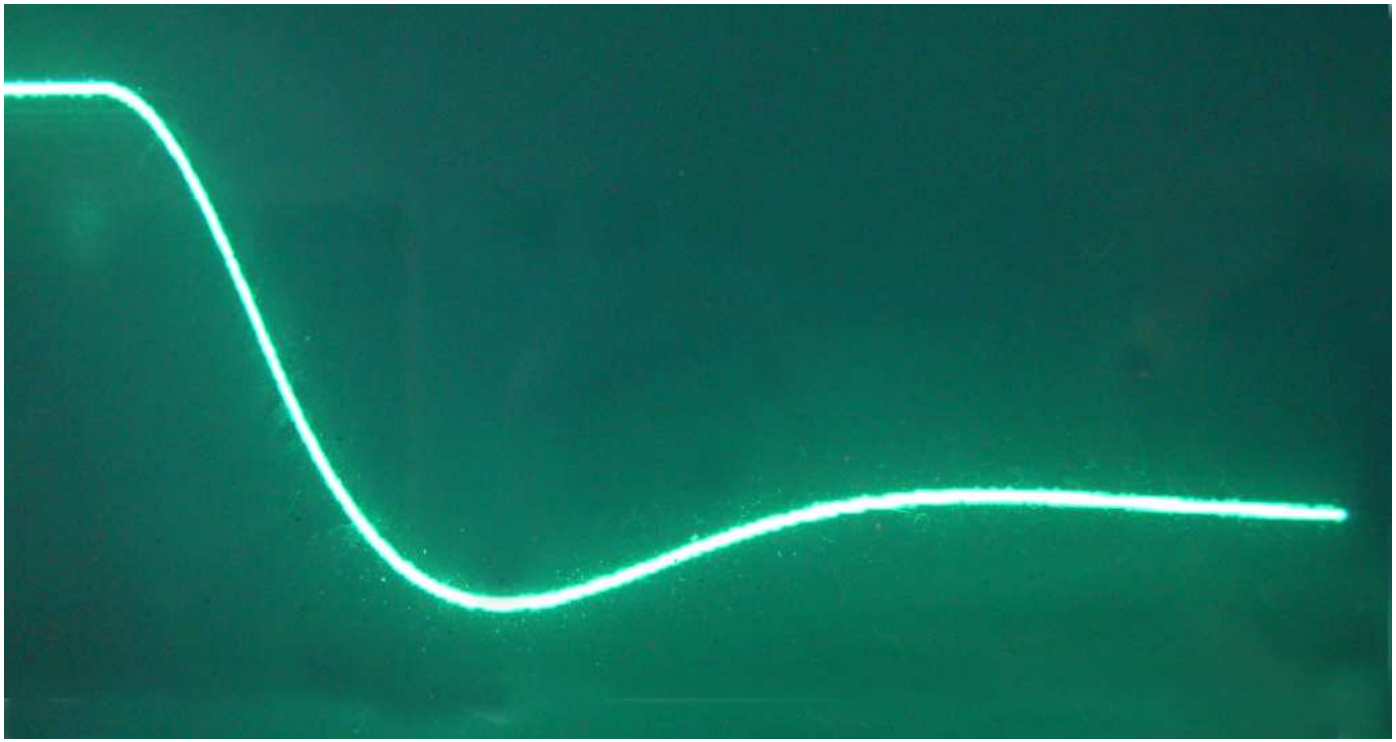


Setting up the computer

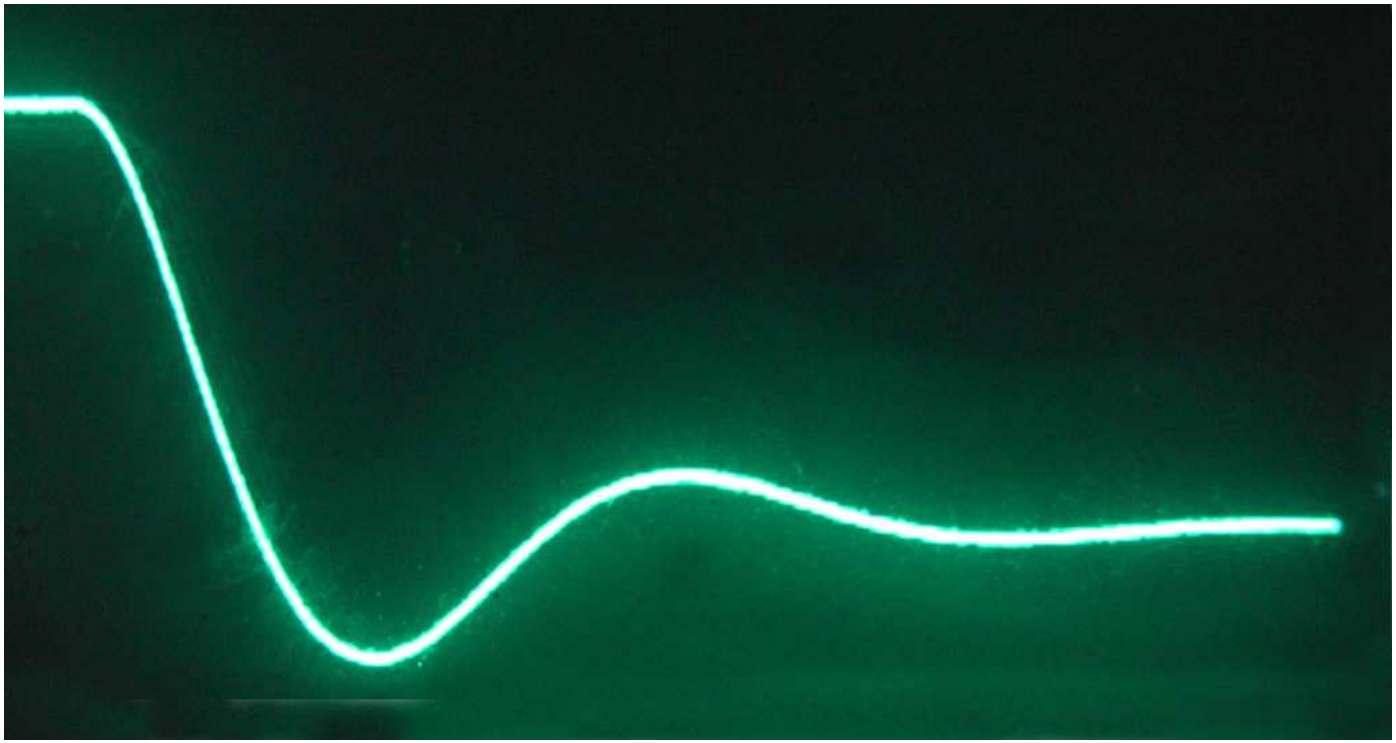


Simulation run with $s = 0.2$ and $d = 0.8$

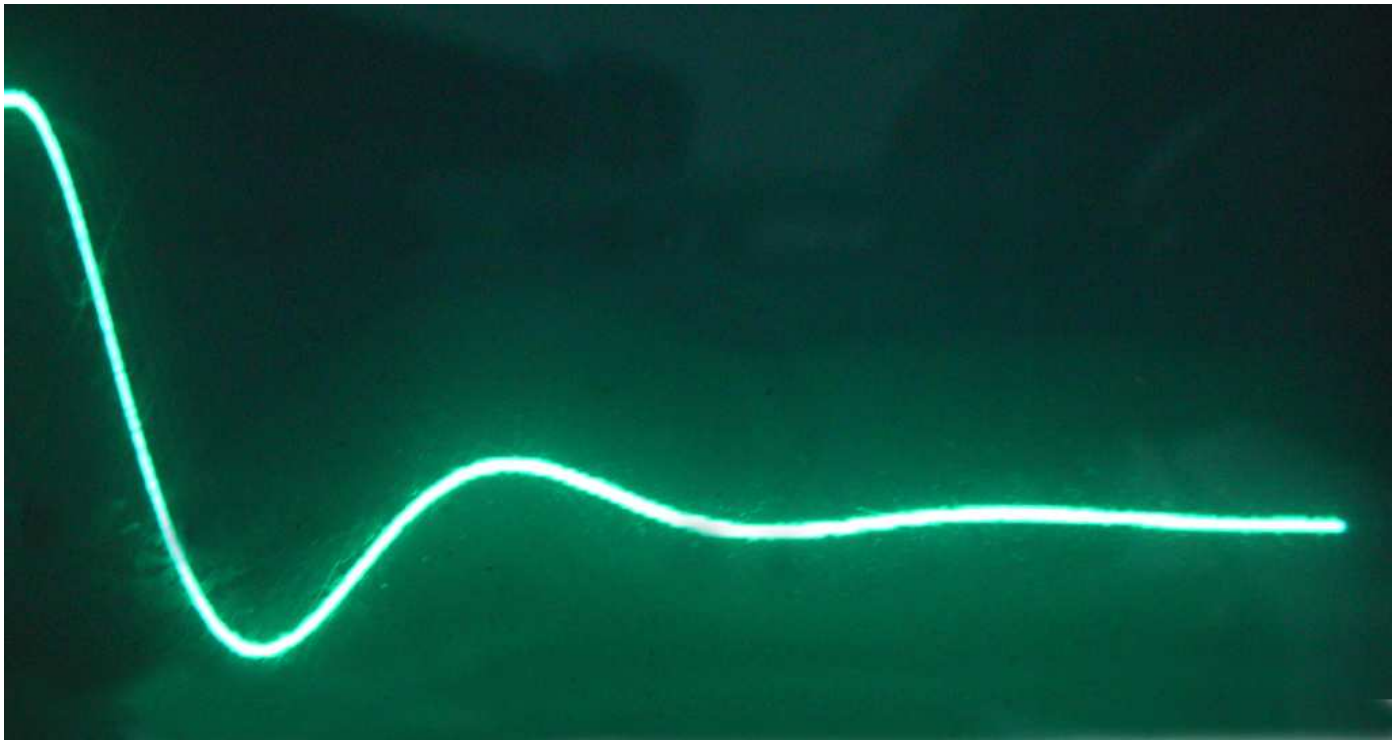
Here and in the following $m = 1$ is assumed.



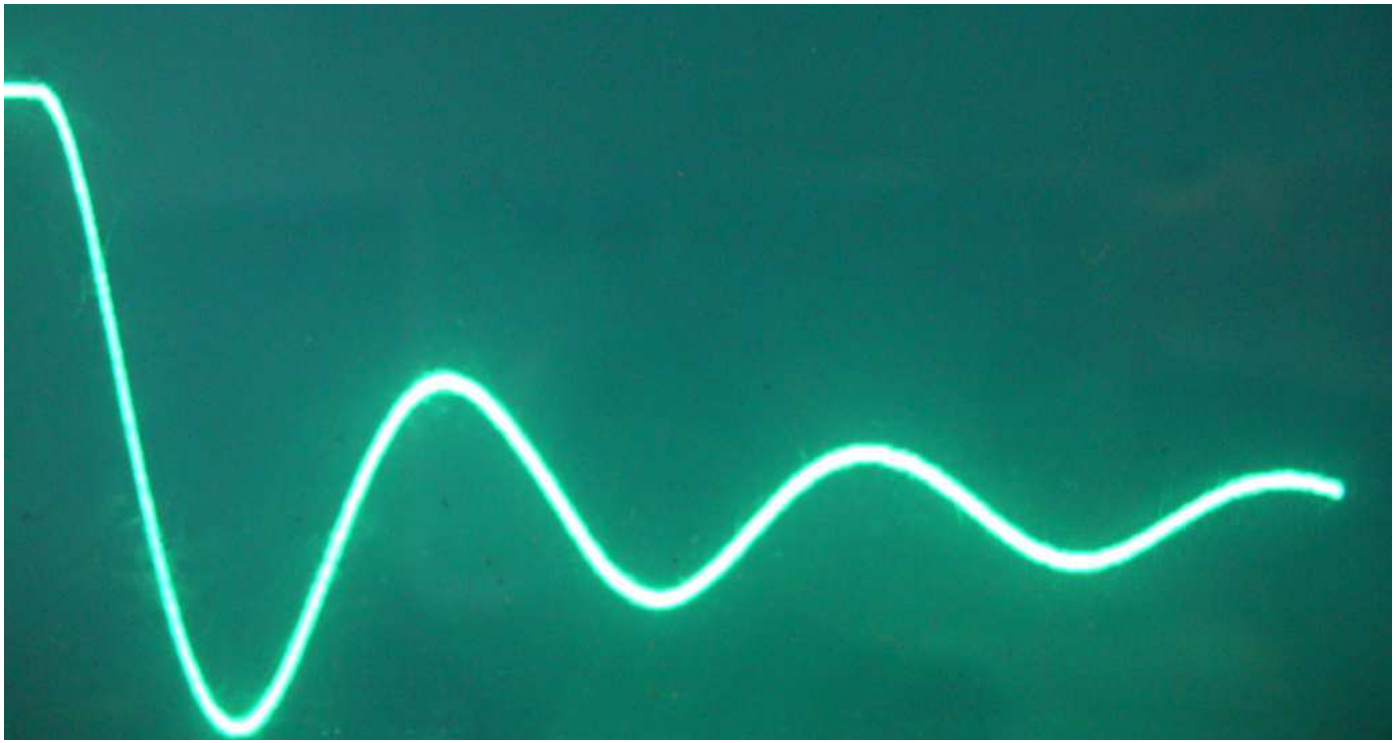
Simulation run with $s = 0.4$ and $d = 0.8$



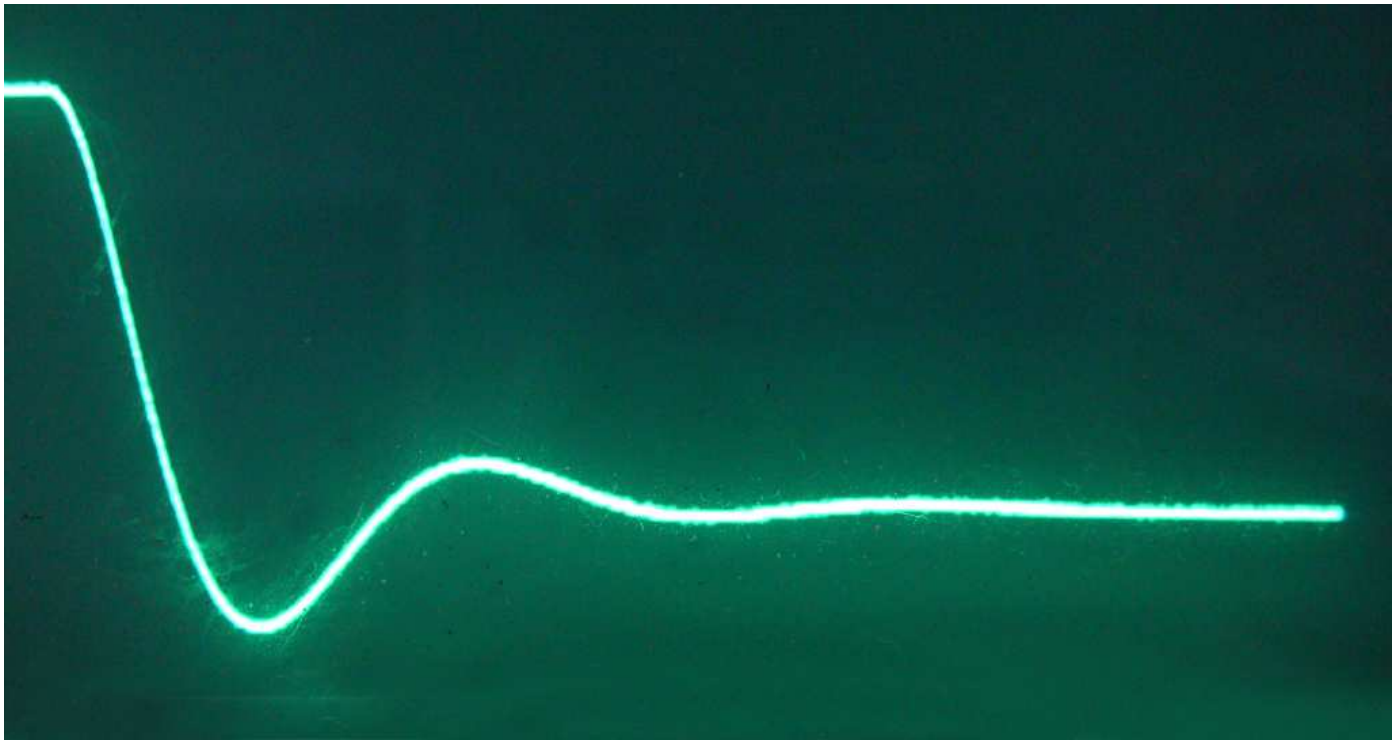
Simulation run with $s = 0.6$ and $d = 0.8$



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



Simulation run with $s = 0.8$ and $d = 1$



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:

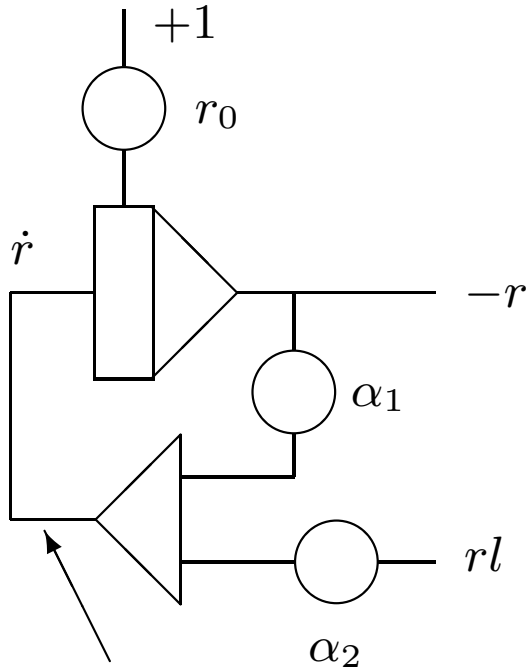
$$\begin{aligned} \dot{r} &= \alpha_1 r - \alpha_2 r l \\ \dot{l} &= -\beta_1 l + \beta_2 r l \end{aligned}$$

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

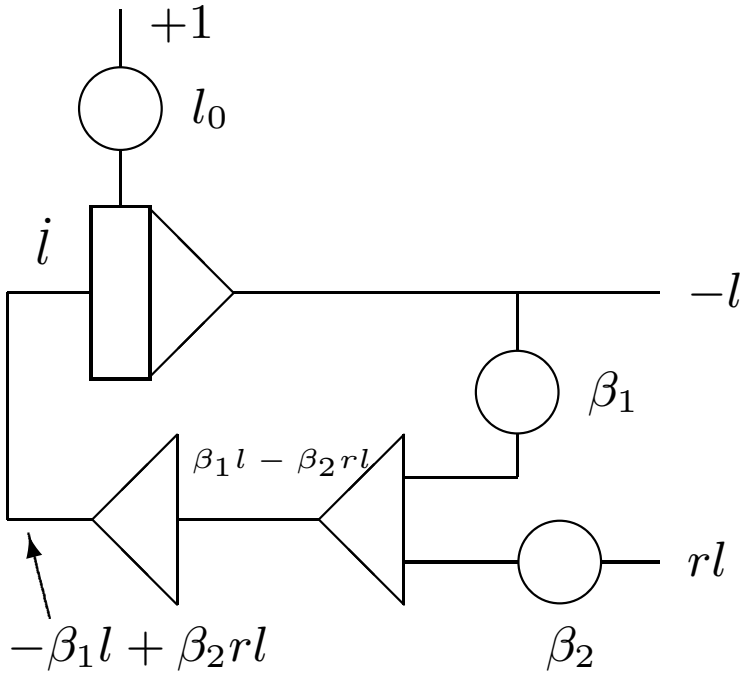
First of all, let us solve $\dot{r} = \alpha_1 r - \alpha_2 rl$ assuming that there is a value rl already known. This leads to the following program:



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

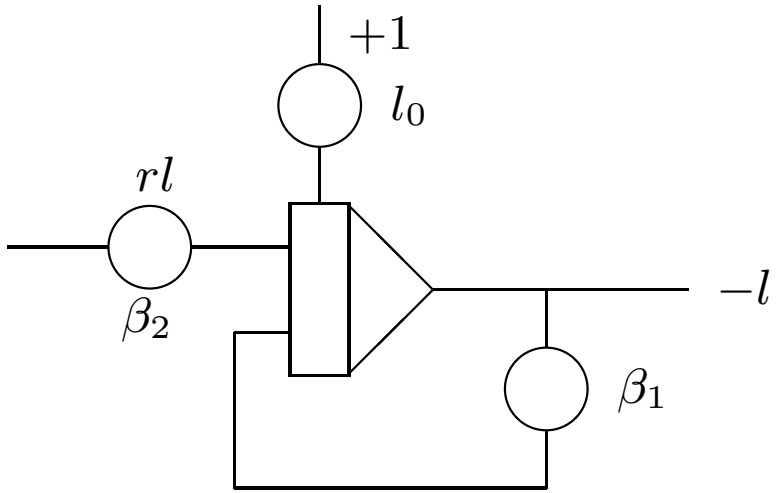
Partial circuit for $\dot{l} = -\beta_1 l + \beta_2 rl$

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

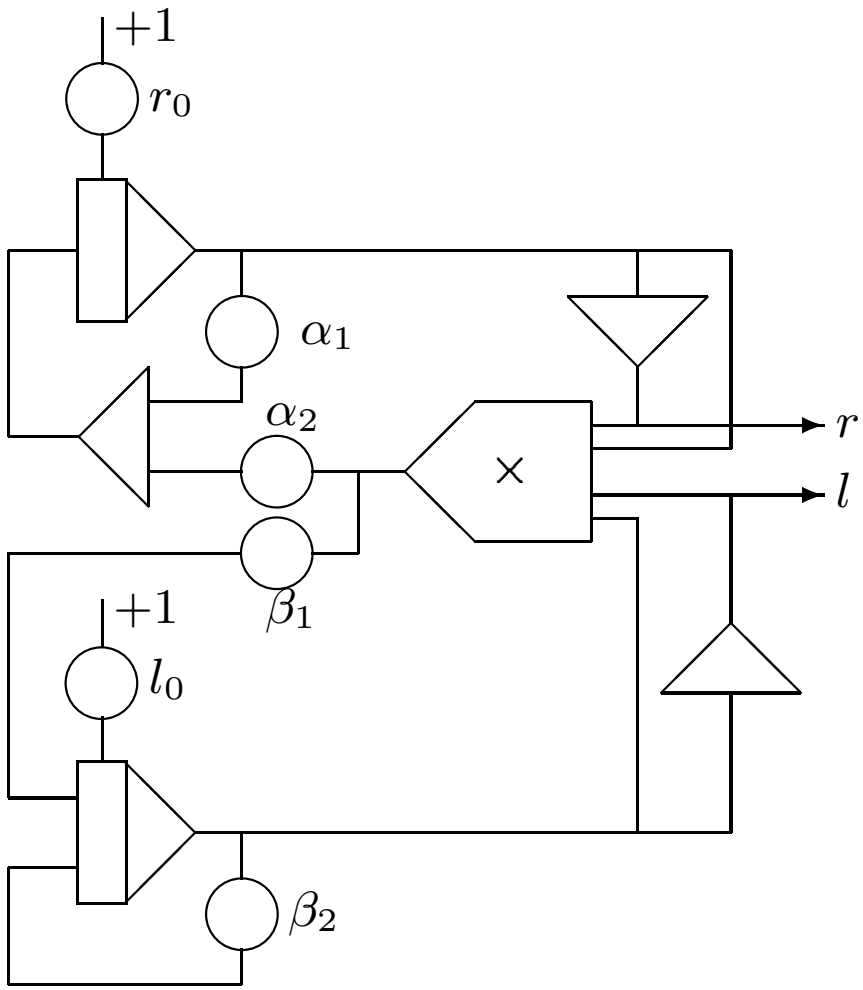


Partial circuit for $\dot{l} = -\beta_1 l + \beta_2 r l$

Obviously we can do better and save two summers:



Coupling both differential equations



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.

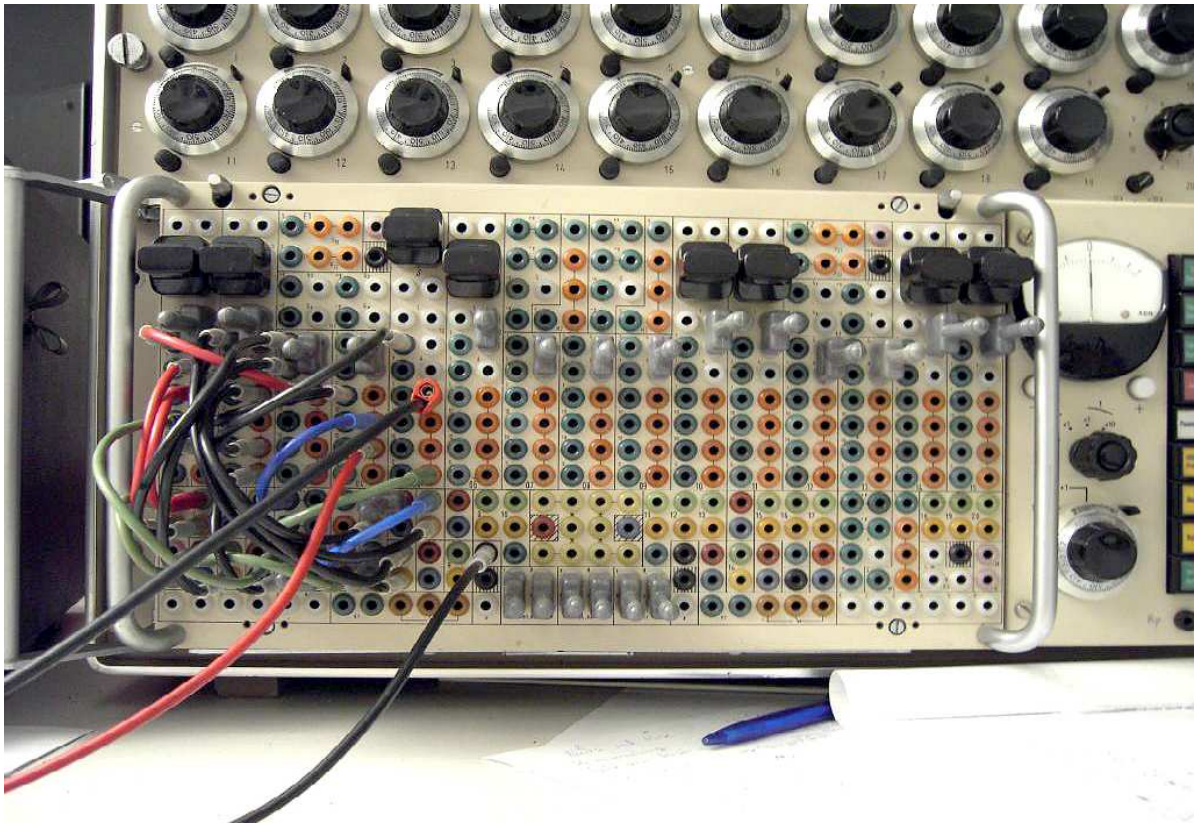
Coupled differential equations like the example above are normally quite difficult to scale since it is challenging to estimate maximum values for the variables.

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).

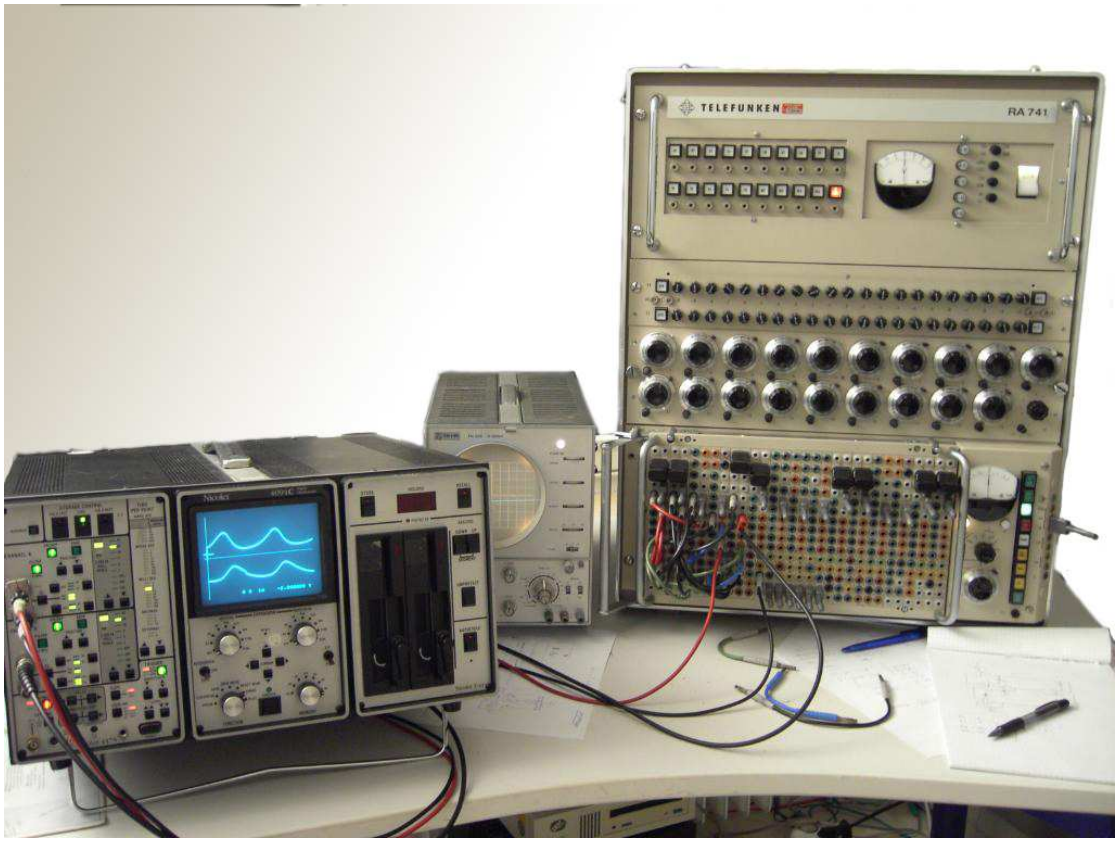
The completed program

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



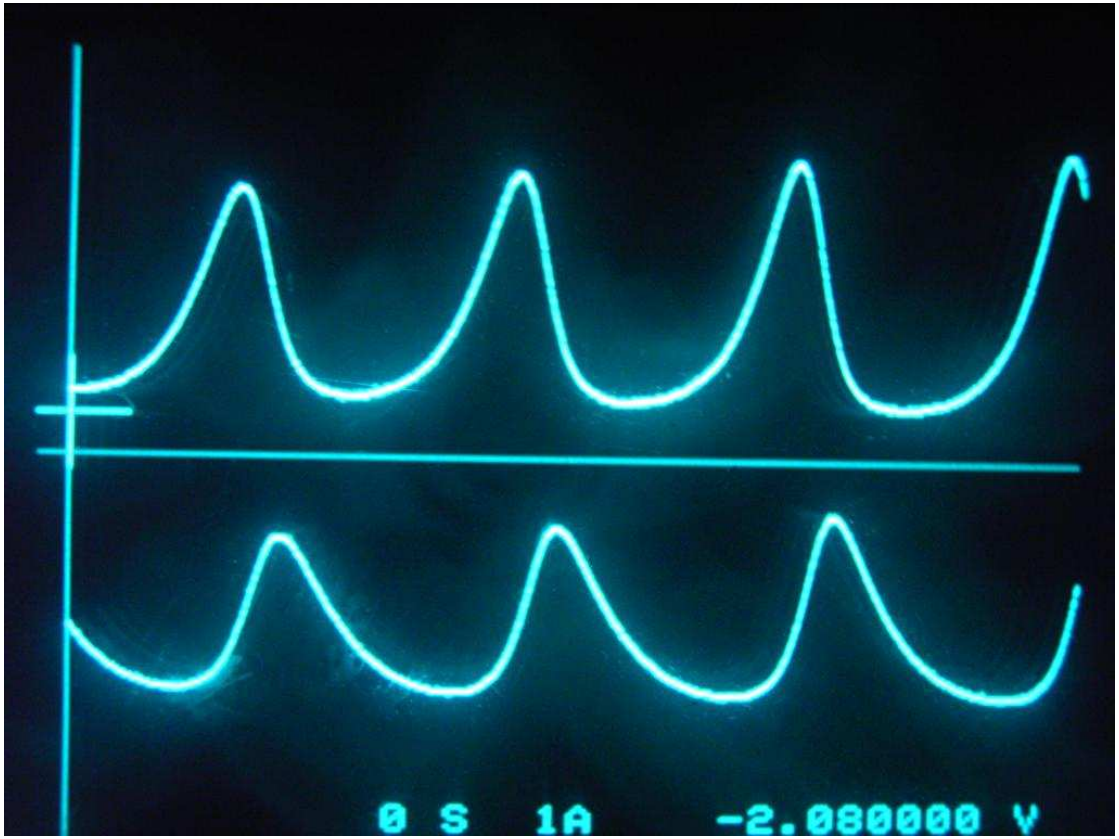
The overall setup

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



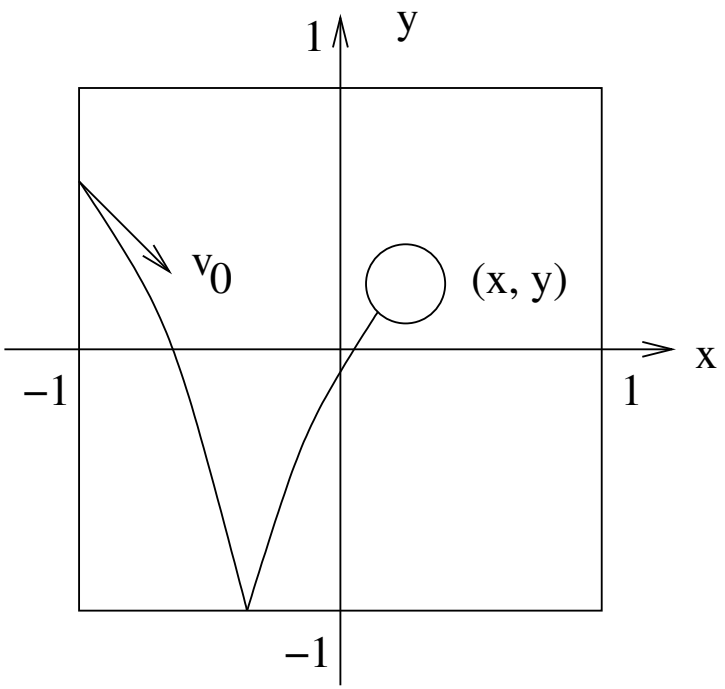
Running the simulation

The picture below shows the results of the running simulation:



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.

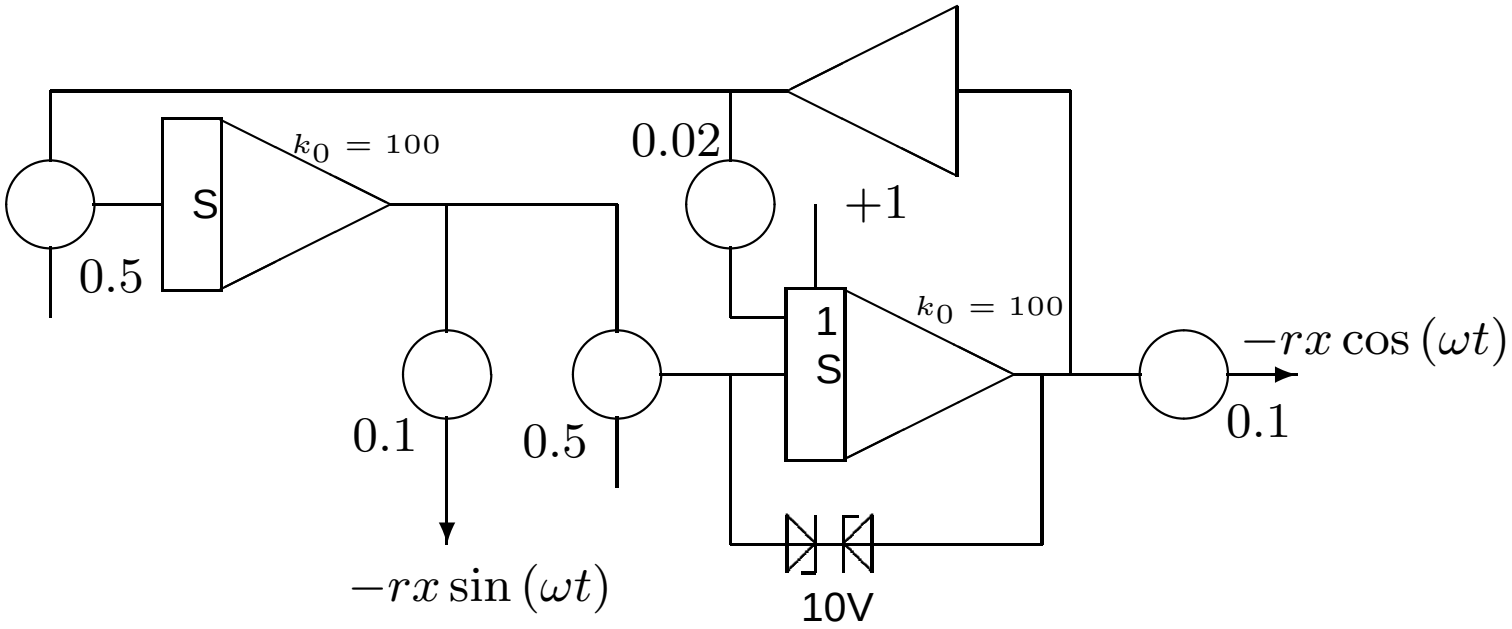
The ball is influenced by a gravitational force pointing downwards and it looses energy by air friction (which is assumed to be proportional to the speed of the ball).

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.

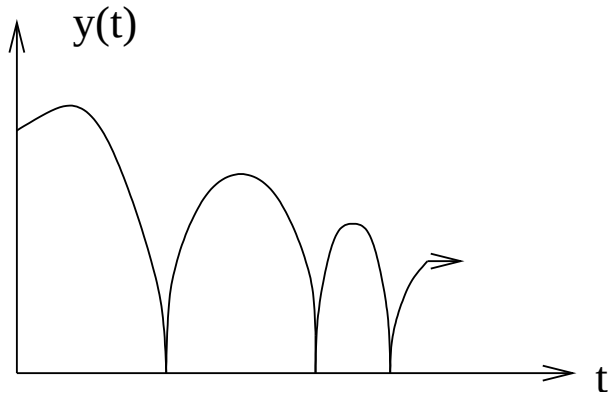
The first thing to note is that the summing junctions of the integrators are used as the main inputs, thus allowing the use of variable input resistances by means of coefficient potentiometers. This is necessary to obtain the desired high output frequency (large ω).

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:

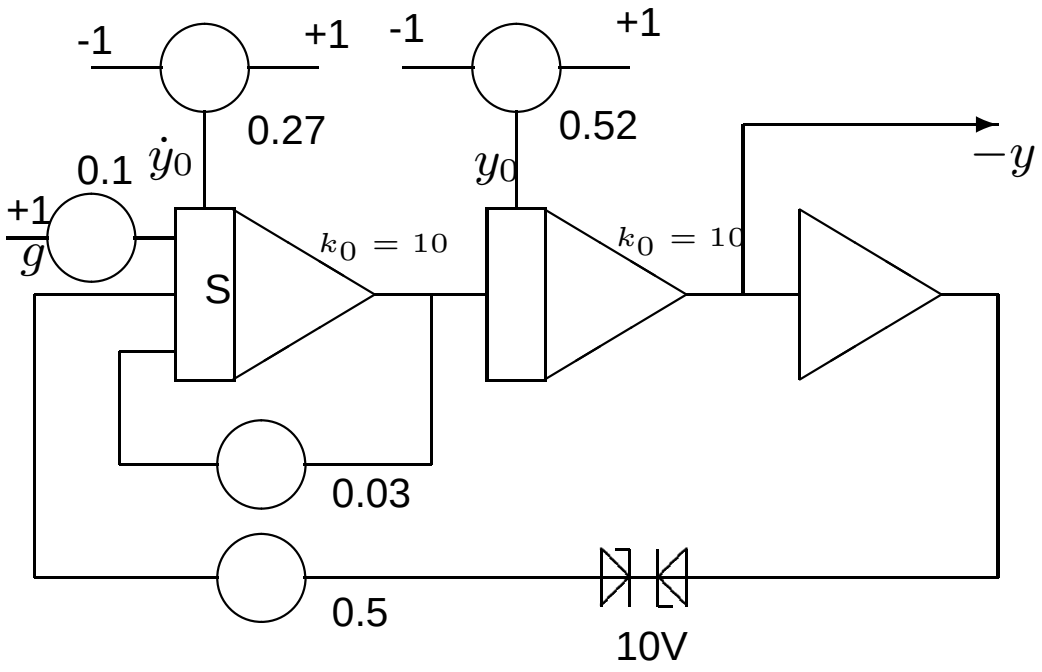
$$\ddot{y} = -g + d\dot{y} \begin{cases} +\frac{c}{m}\left(|y| + 1\right) \text{ if } y < -1 \\ -\frac{c}{m}\left(y - 1\right) \text{ if } y > 1 \end{cases}$$

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

$$\dot{y} = \int_0^T \ddot{y}\, \mathrm{d}t + \dot{y}_0$$

$$y = \int_0^T \dot{y}\, \mathrm{d}t + y_0$$

Computer setup to calculate $y(t)$



Tricks

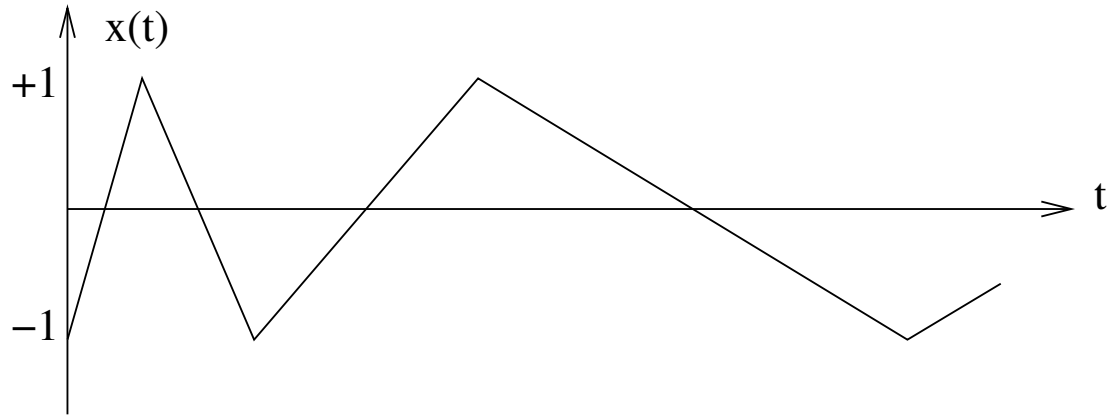
There are some tricks used in this computer setup:

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.

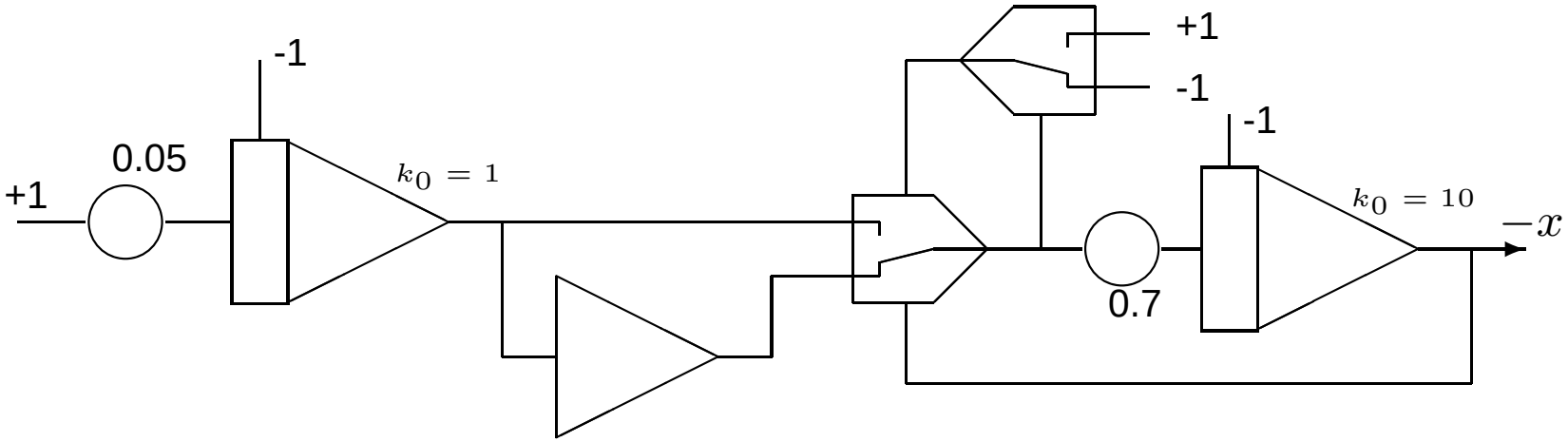
Calculating the x-position

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).



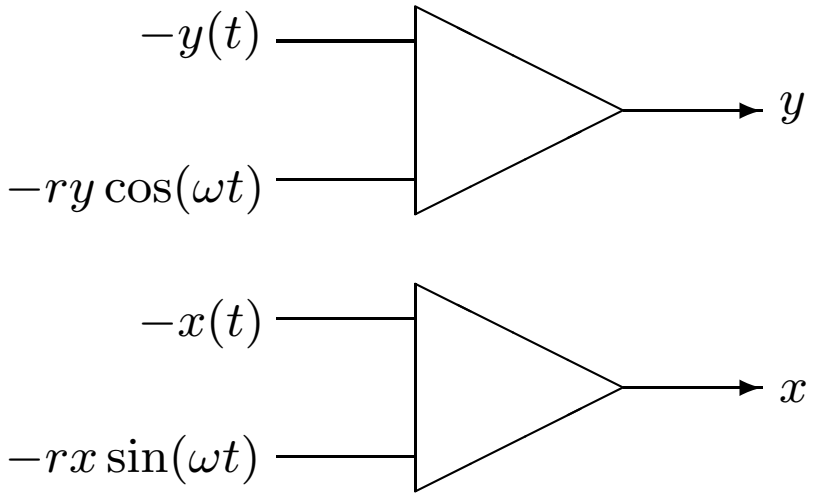
Changing the direction of the ball when it hits the left or right wall is a bit tricky as the following computer setup will show.

Computer setup to calculate $x(t)$

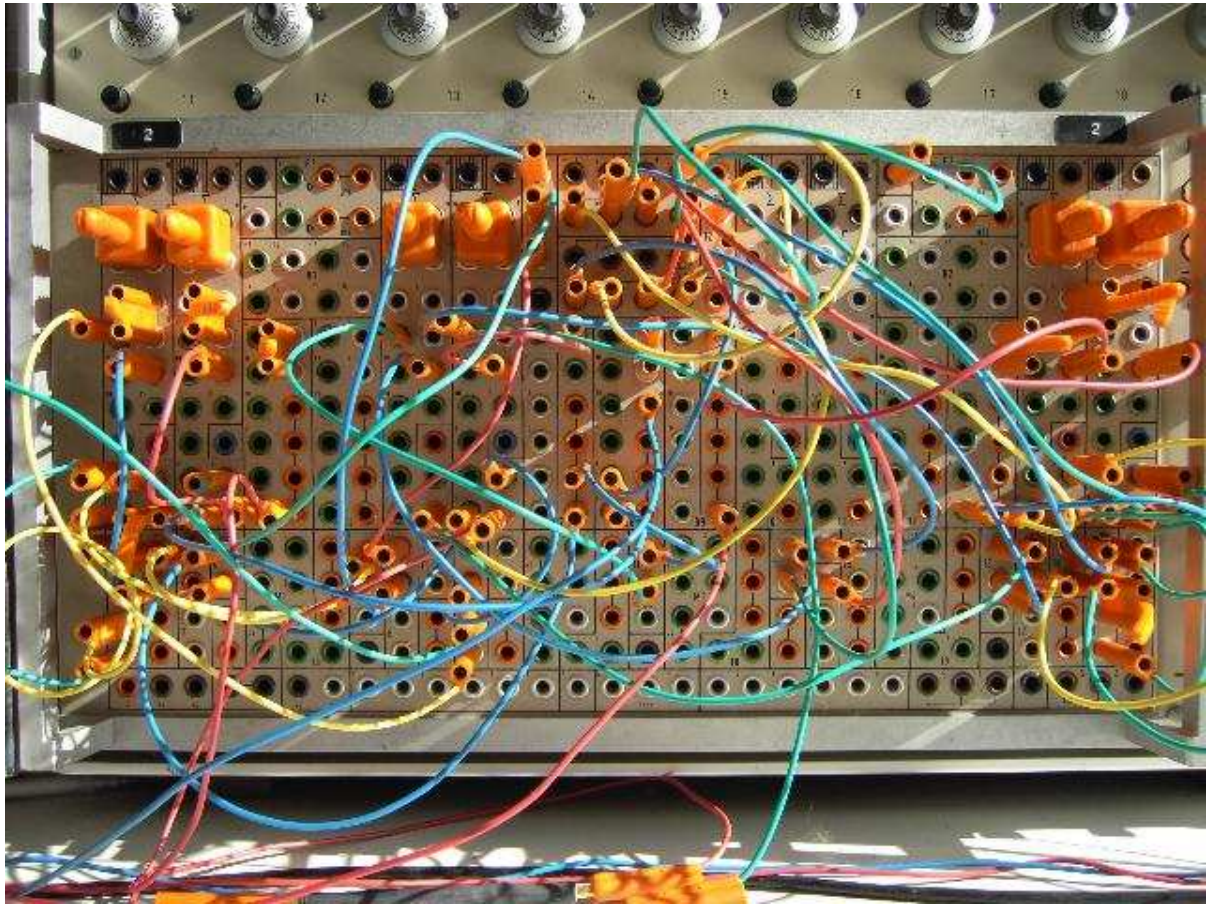


Putting everything together

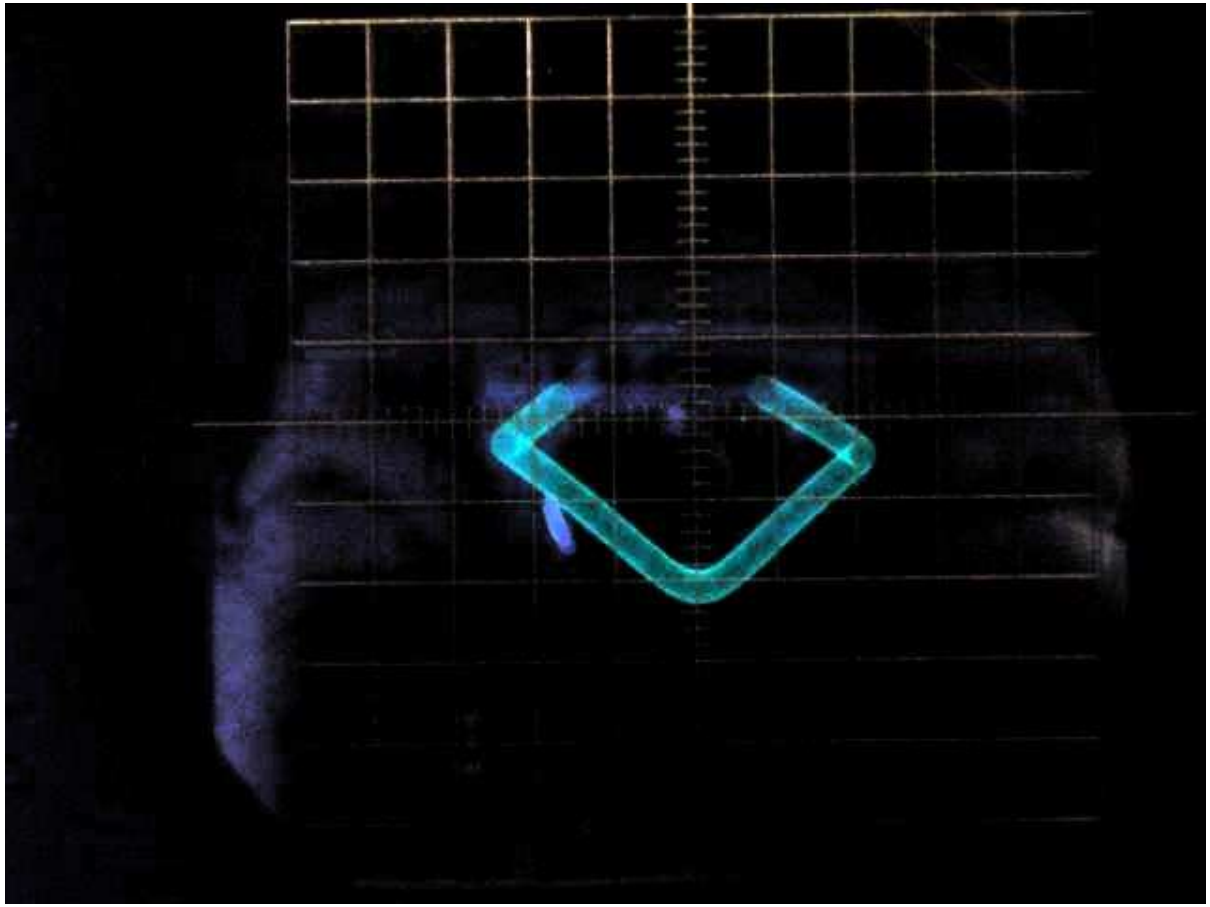
Now having calculated $y(t)$ as well as $x(t)$ all that is left to do is to superimpose those values with the $(\sin(\omega t), \cos(\omega t))$-pair generated previously to display a real ball at a particular position vector $(x(t), y(t))$:



The final computer setup



Bouncing ball



The end

I hope you enjoyed the lecture and maybe you got a bit of the feeling of thinking the ”analog way” as Dr. Giloi once said.

Analog computers are more than just fascinating relics – they are the last reminescences of a wonderful technology and (way more important) they teach one to think in a way completely different from the way one with a background in digital processing is trained to follow.

Thinking the analog way will result in solutions which might never have been thought of in a conventional digital environment.

Thank you for your interest and your patience.

Help

As you may have noticed, analog computing is my passion! Therefore I would like to ask you for help:

- I am trying to save analog computers from scrap whereever I can. If you happen to know about a system looking for a good home, please let me know. (I am not afraid of large systems and I would really love to get an RA770 or an RA800(H), for example! I will pay for all expenses to save the machine from scrap and I will arrange shipping, etc.)
- I am interested in trivia, documentation, computer setups, sales brochures, etc.
- Please spread the word and help to save these machines from getting lost and forgotten.

You can reach me always at `ulmann@vaxman.de` or by mobile phone at 0177/5633531 (in Germany) in case of an emergency. Thank you very much!

References

[1] AEG Telefunken, ”Demonstrationsbeispiel Nr.5, Ball im Kasten”.