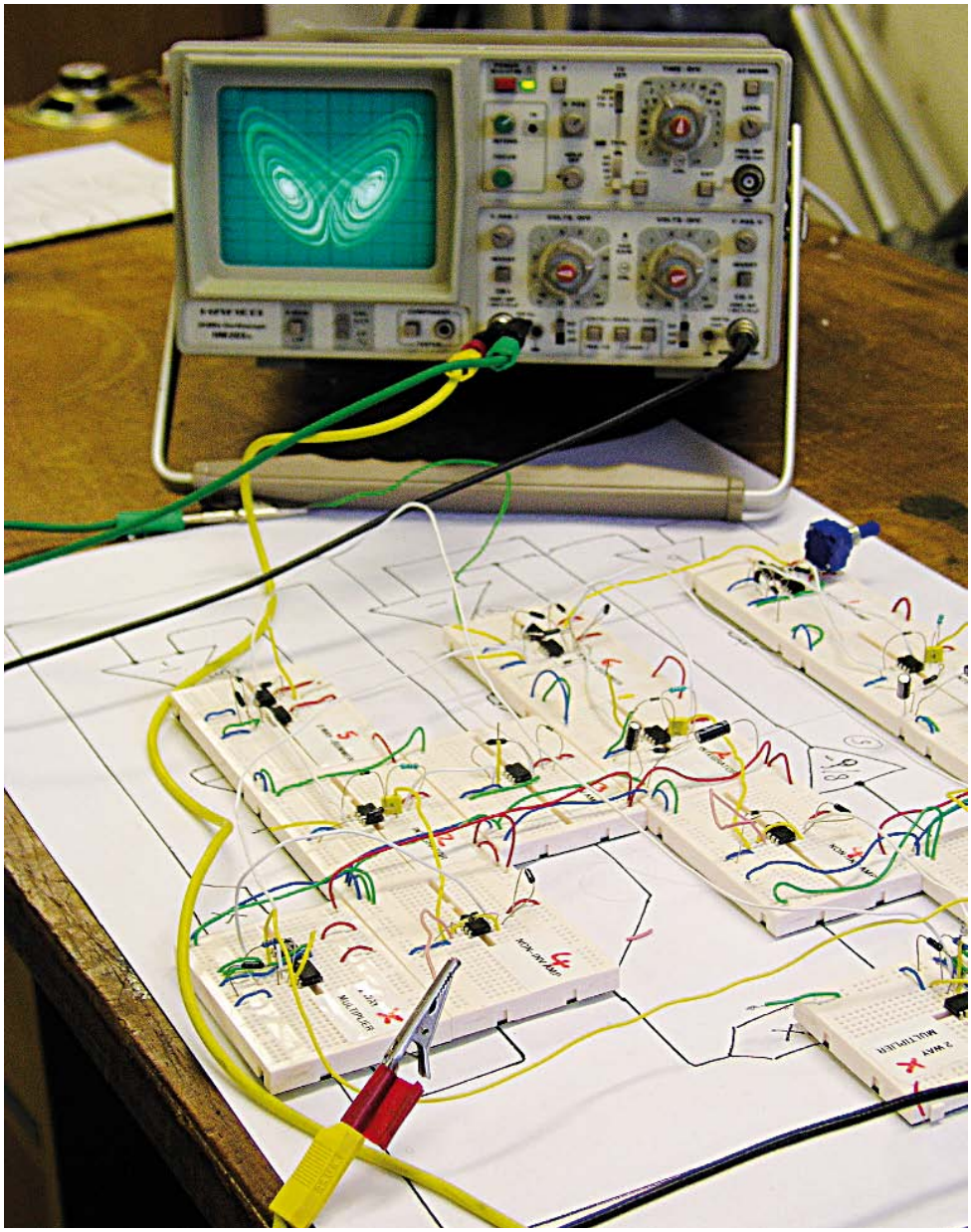
The Chaos Machine

Analogue Computing Rediscovered (1)

By Maarten H. P. Ambaum and
R. Giles Harrison
(Department of Meteorology,
University of Reading, UK)



Analogue computers provide actual rather than virtual representations of model systems. They are powerful and engaging computing machines that are cheap and simple to build. This two-part Retronics article helps you build (and understand!) your own analogue computer to simulate the Lorenz butterfly that's become iconic for Chaos theory. First, however, some history and background.

For some of us it may be surprising that, before the mid sixties, hardly any computing in real-time applications was done by a digital computer. Instead, analogue computers were used because of their speed and relative reliability. Analogue computers are machines that are built to behave as the system we want to compute.
A famous example is the Phillips MONIAC computer from the 1950s [1] (**Figure 1**) which used water flow through Perspex pipes to model the flow of money in an economy. However, in most practical applications, electronic analogies were used. The word analogue refers to the behaviour of the computer being analogous to that of the system we want to simulate. In contrast, the word digital refers to the process of transforming the behaviour of a system to a stream of numbers or digits calculated by a numerical algorithm. Although this is the origin of the word analogue, its meaning has now evolved to describe anything that is not digital.

Modern analogue computers

In the sixties it became clear that the digital computer would rapidly overtake the analogue computer. Advances in chip technology made the digital computer reliable and available to a large num-

ber of people and organisations. However, precisely because of the advances in chip technology driven by the digital revolution, we can now build very cheap and very accurate analogue computers as well.
As part of an art-science collaboration in our Department, we decided to exploit the accuracy of modern analogue electronics in an exhibit of an analogue computer for the *Lorenz model* which produces the butterfly that became the iconic cartoon for the science of chaos and the unpredictability of weather (See, for example, James Gleick's *Chaos: Making a new science* for a wonderful introduction to chaos theory and its history).
Our building of the analogue computer turned out to be an inspiring and illuminating experience. Here we discuss some of the remarkable properties of analogue computers, perhaps no longer widely appreciated. In next month's instalment we will also describe how to make the analogue computer that simulates the Lorenz model. We call it the *Chaos Machine*.

Butterflies; poltergeists; mathematics

The Lorenz equations were developed in 1963 by the meteorologist Ed Lorenz to mimic the flow of air heated from below [2]. They are a set of three coupled equations that describe the time evolution of three variables X, Y , and Z,

$$dX/dt = \sigma(Y-X)$$

$$dY/dt = \rho X - Y - XZ$$

$$dZ/dt = XY - \beta Z$$

The link of these three equations to actual air flow is rather obscure, and they do not work very well anyway.
What Lorenz did discover was that when he chose the three tuneable parameters (σ; ρ; β) in his model carefully it would behave in an erratic and unpredictable way: chaos. This was completely unexpected and paved the way to a revolution in science. If the three variables X, Y, and Z are plotted as a moving point in three-dimensional space, we get the famous Lorenz butterfly, a fractal floating in three-dimensional space, see **Figure 2**.
In our Chaos Machine we can feed two of the voltages that represent the X, Y, and Z, to an oscilloscope in XY-display mode to see it draw an electronic version of the butterfly. We can tune the three variables to produce various shapes of the butterfly. We can also feed the channels to an audio amplifier to hear the sound of chaos. This turns out to be a remarkably unsettling experience: the *Chaos Machine* screeches and screams in the most bizarre ways with the soul of an electronic poltergeist.

How do analogue computers work?

An electronic analogue computer solves equations by representing values of variables by voltages in a circuit. Wires connect modules that perform specific arithmetic operations. For example, a subtraction module will have two input connectors and one output con-

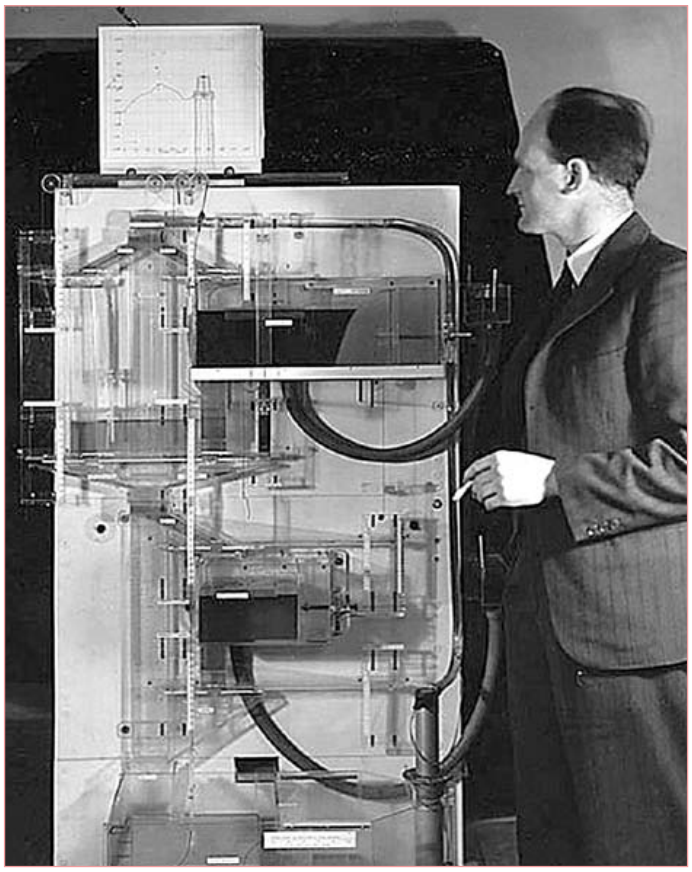


Figure 1. Professor A.W.H (Bill) Phillips was an LSE economist known for the 'Phillips curve' and he developed MONIAC, an analogue computer that modelled economic theory with water flows. *Image:* Wikimedia Commons.

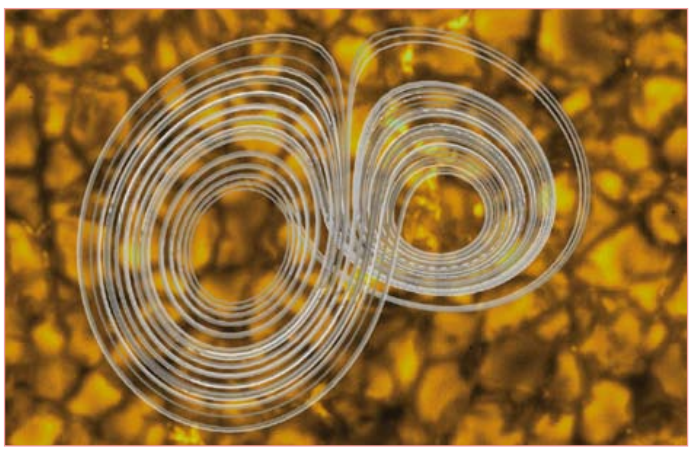


Figure 2. The Lorenz butterfly; the background is an image of solar convection, the original inspiration of the Lorenz equations.

nector where the output voltage equals the difference between the input voltages. This particular module is in fact simply a differential amplifier with unit gain.

The topology of an analogue computer is similar to that of our brain with the axons being represented by the wires, the cell body by the arithmetic modules, and the input ports by the dendrites. Contrast this with a digital computer. In a digital computer variables are stored in memory spaces which are then occasionally operated upon by copying these memory spaces to the central processor which then changes the values of variables in other memory spaces. Digital computers only change values of variables if the central processor says this should happen, and if so, they alter successively. In an analogue computer values always remain consistent. So, if for three variables *a*, *b*, and *c* we have *a*+*b* = *c* then in an analogue computer this will always be the case. There is no internal clock speed; calculations happen instantaneously. In a digital computer this is only valid after the central processor has performed this addition and then only until either *a* or *b* are updated again.

Time integration is also a very natural process for an analogue computer. The input and the output voltages of a time integrating module are always consistently related: at all times the output voltage is equal to the time-integral of the input voltage. There are no time steps involved, as would be the case for a numerical integration routine. Numerical instability of integration routines is not an issue, nor are computing or storage overheads. The basic circuit of an integration module is shown in **Figure 3**. Integration in time occurs by converting a voltage to a current through use of an operational amplifier and then using this current to charge a capacitor. The instantaneous voltage across the capacitor is the time-integrated value of the input voltage. The schematic shows an electronic circuit able to perform integration in time of a varying input voltage. It is based on two operational amplifiers A1 and A2 each having inverting (–) and non-inverting (+) inputs and an output terminal. A time varying voltage $V_1(t)$ is applied to A1, which drives the integrator circuit comprising *R*, *C* and A2. The output voltage $V_2(t)$ is (minus) the time integral of the input voltage, scaled by (1/*RC*). A1 is a unit gain buffer stage, included solely to prevent loading of the originating voltage source but permitting a wide choice of values for *R*. (The additional resistor with A2 is for compensation and does not form part of the functional circuit.) For clarity, the necessary power supplies are not shown.

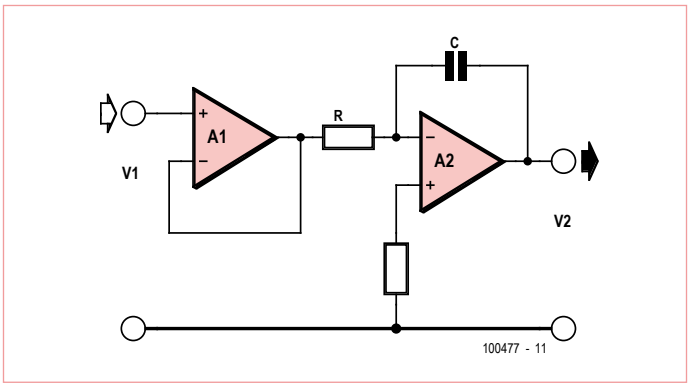


Figure 3. Basic circuit of the time integration module.

Other arithmetic operations can also be performed with the help of operational amplifiers. For example, subtracting two voltages is achieved using a differential amplifier of unit gain. Multiplication and other related operations are more complicated to implement, requiring many op amp stages.

Analogue computers do not require a memory to work. This makes them essentially equivalent to the systems we try to simulate. A swinging pendulum does not have a memory of its previous states. One could connect an analogue computer to analogue–to–digital converters if digital storage or exact measurements are required. This construction can also be used to build a hybrid analogue–digital computer. A purist who wants to stay away from any digital technique can use a tape recorder or chart recorder for storage, also circumventing the difficulties of aliasing which arise in a sampled system.

Analogue computers are relatively hard to program: programming the computer is the same as building the computer. Clearly this flexibility is where digital computers are far superior. Also, in a digital computer it is easy to allocate memory spaces to store a set of variables, while in an analogue computer each variable is associated with a separate signal wire. Although a digital computer requires much more complex hardware for variable storage, it can use the same hardware configuration to tackle different virtual problems. A digital computer is a universal Turing machine, that is, a machine that can be used to solve different problems; an analogue computer can only solve one single problem.

Another fundamental difference between analogue and digital computers is that a digital computer calculates an approximated virtual representation of the model system, whereas an analogue computer is an actual electronic copy of the system. If we want to simulate a swinging pendulum with an analogue computer, we build an electronic system that oscillates exactly like the swinging pendulum. The computer becomes an electronic version of the swinging pendulum itself. This is a very appealing property of analogue computers. Think of the Lorenz system that we use in our *Chaos Machine*. Apart from a very artificial set-up, there is no actual physical representation of the system; it was designed as a mathematical system. Analogue computers are the only way we can get genuine physical representations of such mathematical systems.

How fast are they?

People who see an analogue computer for the first time often ask: how fast is it compared to a modern digital computer? In fact, their speeds are hard to compare. In a digital computer speed is limited by the clock speed of the processor and the speed at which variables can be loaded into and out of the processor. One such calcu-

lation may typically take a nanosecond or so (one thousand millionth of a second). In an analogue computer the speed is limited by the speed at which the operational amplifiers, the key building blocks of analogue computers, can follow changes in input voltages (the *slew rate*). Operational amplifiers can change over time scales of a few nanoseconds and for most practical purposes this does not limit the computer's operation. However, analogue computers do not perform calculations as such; they perform simulations. Asking how fast an analogue computer calculates is the same as asking how fast the swinging pendulum calculates its motion.

Nevertheless, the 'speed' comparison can be made more precise. An analogue time integration module has an intrinsic timescale set by $R \times C$, the resistance and capacitance, respectively, of two components in the module. In other words, speed in a digital computer is limited by its clock speed, while 'speed' in an analogue computer can be arbitrarily defined by choosing different components. There is a practical limit to which we can increase the speed of the analogue computer set by the finite slew rate of the operational amplifiers and the stray capacitances in the system, both of which act to damp away the very highest frequencies.
Fortunately, dedicated analogue function chips are cheaply available which contain optimized log and antilog converters, providing multiplication, divisions, and square roots with excellent accuracy and temperature stability — and truly enormous speed.

Operational amplifiers

Advances in electronic components have had major benefits for many scientific activities, but they also now make the implementation of analogue computing straightforward. The key building block of an electronic analogue computer is the operational amplifier, a general purpose electronic device which can be configured to perform the different mathematical operations (integration, addition, multiplication, scaling) required. General purpose amplifiers originated in the 1940s from military applications, particularly in anti-aircraft gunnery (although a mechanical analogue computer was used as recent as the Vietnam war in the 'Norden' bombsight to target bombs dropped from aircraft). The description *operational amplifier* (or *op amp*), appeared in 1947, and the first commercial op amp — type K2-W — was produced by Philbrick in 1953, based on two dual triode valves, see **Figure 4** and [3,4]. Solid state op amps followed in the 1960s, with the first integrated circuit op amp in 1965 (Jung's *Op Amp Applications Handbook* provides a good overview of the history and use of op amps).
Comparison between early and modern op amps illustrates how the steady improvements have made analogue computing ever more practical. The thermionic K2-W had a specified drift of ±5 mV per day, whereas the integrated circuit OP97, used in our Lorenz design, has drift dominated by thermal changes, at 0.6 µV/°C. The power requirements are also dramatically different. A K2-W required power supplies of ±300 V and 6.3 V, at about 10 mA and 0.6 A, whereas the OP97 requires ±15 V at 0.6 mA. Early analogue computers therefore had a substantial physical volume associated with each computing stage, as well as power dissipation and heat generation.
Integrated circuit analogue computers are now compact, and drift is no longer a characteristic feature. Individual op amp stages are also relatively cheap, allowing complex systems to be readily simulated. A benefit of their low cost is that extra stages can easily be included, which although not essential to the computing function, may relax constraints on the components required.

Figure 4. The Philbrick K2-W is generally considered the first commercial operational amplifier.

The final analogue computer is an assembly of independent circuit modules, combined to solve one specific problem, but reusable for other applications. With such a modular approach the 'programming' of the computer is a fairly simple job which requires hardly any knowledge of the electronics involved.

Next month's second and final instalment discusses the elements that go into building the *Chaos Machine*.

(100477)

Internet Links and References

[1] http://en.wikipedia.org/wiki/MONIAC_Computer
[2] http://mathworld.wolfram.com/LorenzAttractor.html
[3] www.philbrickarchive.org/
[4] Philbrick K2-W, the mother of all op amps, Elektor (Retronics) October 2009.

Retronics is a monthly column covering vintage electronics including legendary Elektor designs. Contributions, suggestions and requests are welcomed; please send an email to editor@elektor.com

The Chaos Machine

Analogue Computing Rediscovered (2)

By Maarten H. P. Ambaum and R. Giles Harrison (Department of Meteorology, University of Reading, UK),
Jan Buiting and Thijs Beckers (Elektor Labs)

The analogue computer we set out to describe in the previous instalment was constructed from separate computation modules for multiplication, integration, summation and scaling, combined to represent the Lorenz 1963 equation system (ref. part 1). The circuits for the modules are largely based on suggestions in Peyton and Walsh, *Analog Electronics with Op Amps: A Source Book of Practical Circuits*, where more details on their functionality can be found. We found the use of breadboards very suitable for this project but have also made a soldered version that travels better.

Modular approach to Chaos

Figure 1 provides an overview of how the computational modules are combined, in terms of the signal paths and **Figures 2a through 2g** provide individual circuit schematics for each of the computational modules required. A summary of their function is given below, but first the

Block diagram (Figure 1). This figure shows the combination of computation modules required for the complete analogue computer. Triangles represent function modules, each with a set of inputs and a single output.

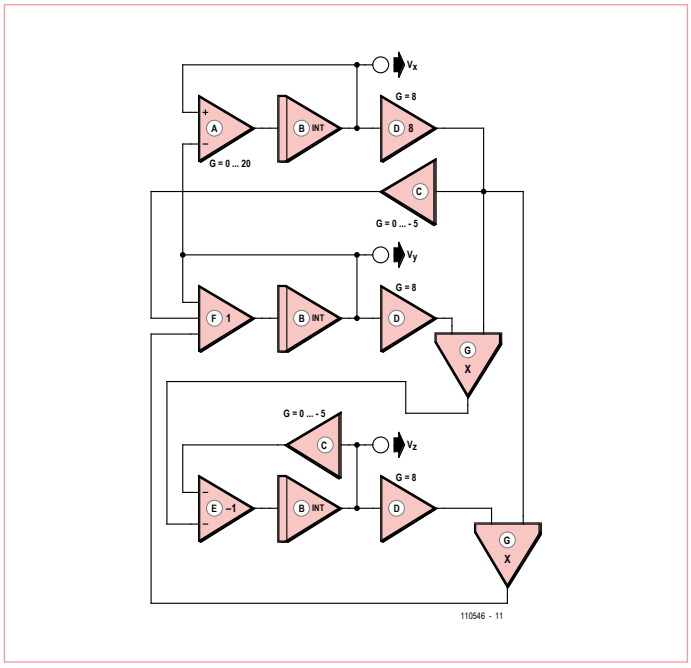


Figure 1. Block diagram of the Chaos Machine. Each function shown corresponds to one of seven basic circuits from Figures 2a through 2g, solving the Lorenz equations.

Symbols (+) and (–) are used to denote non-inverting and inverting inputs, and (×) multiplication. A thin rectangle on the input side of the triangle denotes an integrator. The gain-8 scaling amplifiers (ident: 'D') are included to ensure that the voltages in each wire do not exceed the stated maximum amplitude of ±10 V in any of the op amp input stages and the multiplier chip.
Output voltages are available at the three nodes marked V_x, V_y, V_z, for use with an oscilloscope having an 'xy'-display mode and 'z' axis (intensity) modulation.
This model has a control (in module 'A') to vary the Prandtl number (parameter σ in the Lorenz equations) between 0 and 20, to display the different regimes of the Lorenz equations.

A: Differential Amplifier (Figure 2a). Input voltages V_1 and V_2 are buffered by dual op amp stage A1, subtracted in op amp stage A2.A, and then amplified by an inverting amplifier stage A2.B with gain G (up to × –20). Function: $V_{out} = G\,(V_1 - V_2)$.

B: Inverting Integrator (Figure 2b). Input voltage V_1, referred to the signal ground, is buffered by op amp stage A2.A, and then integrated by stage A2.B. A dual op amp package provides both amplifiers. Function: $V_{out} = \int V_1\,dt\,/\,(3.3 \times 10^{-4}\ \mathrm{s})$

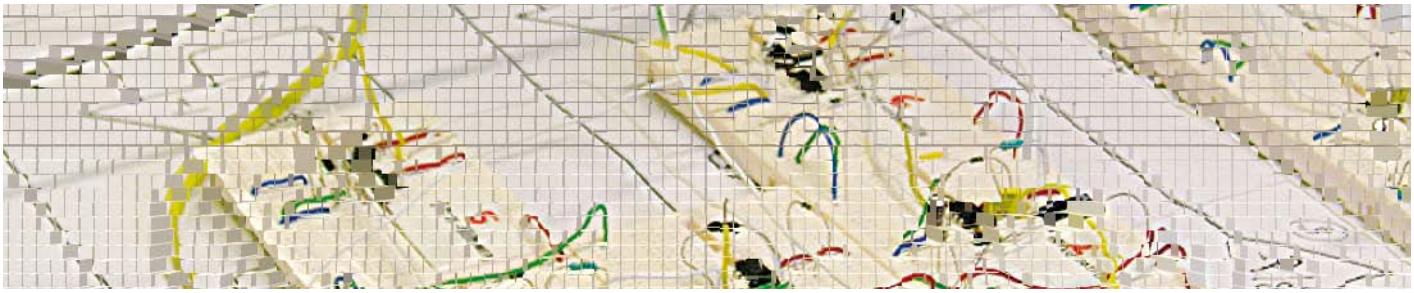
C: Inverting Scaling Amplifier (Figure 2c). Input voltage V_1, referred to the signal ground, is buffered, and applied to an inverting amplifier stage with variable gain G (up to × –5) set by the 100 kΩ potentiometer. Function: $V_{out} = -GV_1$. Top C: $G = -3.5$; bottom C: $G = -2.7$.

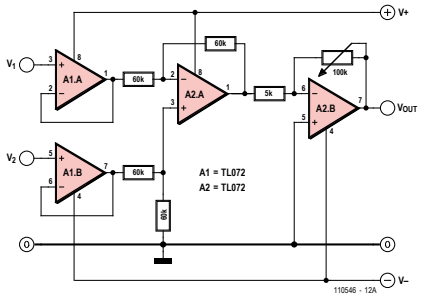
D: Non-inverting Scaling Amplifier (Figure 2d). Input voltage V_1, referred to the signal ground, is buffered, and amplified by a non-inverting stage with a fixed gain of 8. Function: $V_{out} = 8V_1$.

E: Inverting Summer Amplifier (Figure 2e). Input voltages V_1 and V_2 are buffered, and added in the third, inverting, stage. A dual op amp package provides both input amplifiers, and a further package, the summation stage. Function: $V_{out} = -(V_1 + V_2)$.

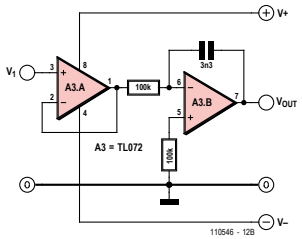
F: Non-inverting Summer Amplifier (Figure 2f). Each of the three input voltages V_1, V_2, and V_3 are buffered, and then added in a summation stage. Two dual op amp packages can be used. Function: $V_{out} = V_1 + V_2 + V_3$.

G: Multiplier (Figure 2g). A function chip (type AD633) is used to determine the product of two input voltages, with a further non-inverting stage contributing a gain of ×10 to establish a scaling voltage V_0 of 1 V. Function: $V_{out} = V_1 \times V_2 / V_0$.

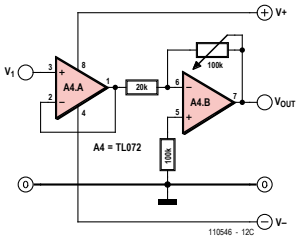




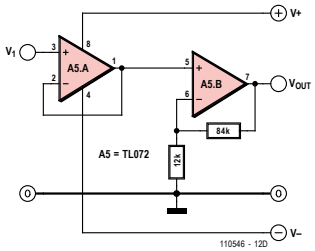
2a. DIFFERENTIAL AMPLIFIER



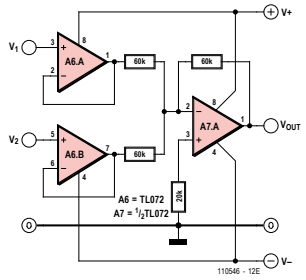
2b. INVERTING INTEGRATOR



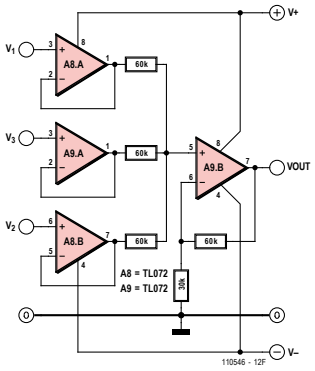
2c. INVERTING AMPLIFIER



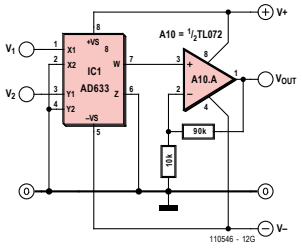
2d. NON-INVERTING AMPLIFIER



2e. INVERTING SUMMER



2f. NON-INVERTING SUMMER



2g. MULTIPLIER

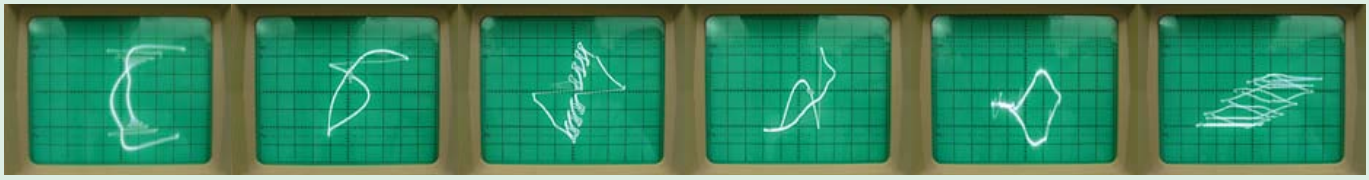
Figure 2. Overview of all required mathematical functions required for the Chaos Machine, realised using op amps for the most part.

Chaos in theory

The basic conclusion of Lorenz's work is that even if you know the initial conditions quite precisely the error in specifying the initial condition rapidly grows, so that after even a short time we cannot predict the *details* of the motion — the sensitive dependence on initial conditions is one of the defining properties of chaos.

Chaos for real

By Jan Buiting, Editor.



Retronics instalments normally meet with silent understanding and approval from Elektor's technically inclined people, and mild surprise or the odd chuckle from all other staff seeing and hearing vintage equipment hauled into and out of the damp cellars of medieval Elektor House. However when the word was out that "*Chaos has descended upon Jan's pages in the September 2011 edition*" many staff were disenchanted to see a neatly formatted Retronics instalment with solid content, and no chaos or other disordered mess to revel at.

Keen to experience chaos for real, a few enthusiasts murmured and then suggested to actually build the Chaos Generator and so we did, where 'we' ≈ [Thijs Beckers ± Jan Buiting] of *Chaosfree Desks*, a small, quiet faction within the Elektor editorial and lab bunch.

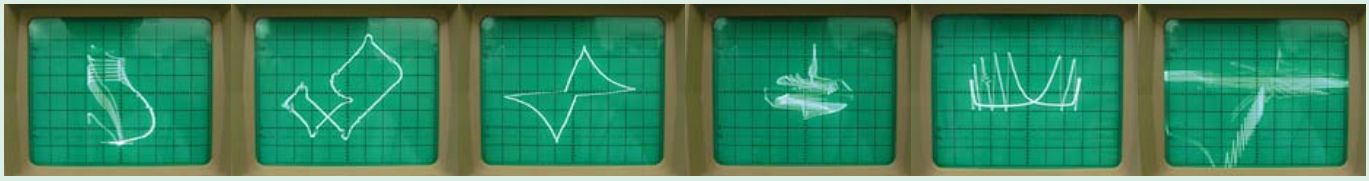
The math functions that hopefully enable the generator to behave chaotically were linked to circuits, components and eventually, modules for interconnecting with wires (signals as well as power supply). The thing worked spot on, producing bizarre images on our Hameg oscilloscope in x-y mode (sadly, all models with z modulation were on the blink). Turning the two pots and occasionally introducing stray capacitance with our fingers under some boards, we were able to produce extremely complex shapes ranging from sea horses to Möbius' bands, DNA strings, styled epsilons and even business models not unfit for graduating Hons. at the *London School of Economics* (LSE) we told our MD, accounts and marketing staff. Dilbert and Professor Bill Phillips would have loved it. For example, advancing the pot on module 'A' (LSE: "upping XYZ Corp.'s sales resources") beyond a certain critical level caused the entire riotous scope image (LSE: "this highly creative organisation") to change shape (LSE: "come to terms with its budgets"), then DC–shift off the scope screen (LSE: "go in pursuit of other challenges") and finally bounce back on to the screen looking like a violent vortex not unlike that in an aircraft toilet (LSE: "sudden depreciation of assets"). As fickle as modern financial markets!

Some of the better chaotic images are shown in this inset, as well as filmed for a video clip you can view on Elektor's very own YouTube Channel (details to be announced online). All of the the images shown here are likely due to opamp saturation at some point in the system.

The sound signals taken from the outputs were as impressive and uncanny as Maarten and Giles mentioned in part 1 of this article. You have to hear it believe it and everyone's invited to check it out at the Elektor Live! event this November.

To some the generator is a gizmo you just can't resist tweaking and adjusting by means of the two controls for yet more wacky shapes on the 'scope screen. To others, it is a serious implementation of complex mathematical functions a powerful DSP of PIC micro would be challenged to produce equal results, i.e. visually and at speed. Admittedly the practical use of the machine is limited at best, with some consolation that the weather is the largest recognised chaotic system we know — with some workplaces at Elektor House happily contending for second place.

At this point, we challenge our readers to investigate if their powerful 32- and 64-bit electronics simulation programs and PCs are up to the disarmingly simple Chaos Generator circuitry shown here. Failing that, or tired of error reports popping up (*Division by Zero!*), send us the best application of the Chaos Generator you can think of! Or a Chaos applet for the iPhone or Android allowing top ranking business people to use it on the train — there's money to be made.



live! chaos

Experience the Chaos Generator at Elektor Live! 2011

Elektor Audio & Retro Division

Eindhoven, The Netherlands, November 26

Suggestions for construction

At Reading University, the components for the prototype were assembled on solderless breadboard, using a separate breadboard for each functional module. All the circuit stages were powered from a common ±15 V power supply, with a conventional ground voltage of 0 V throughout. The operational amplifiers used were type OP97. This op amp is available in single (OP97) and dual versions (OP297), which can be combined to reduce the number of integrated circuit packages required, whilst preserving the independence of the modules. Pin numbers refer to the standard dual in-line integrated circuit packages for the OP97 and OP297. The multiplication stages use a mathematical function chip, the AD633, which uses the same bipolar power supplies.

The replica of the Chaos Machine built at Elektor Labs uses TL072 op amps throughout for the simple reason that they happened to be available. Also, some of the theoretical resistance values like 60 kΩ were replaced by their nearest real-life equivalents lurking in component drawers. To add some aesthetics to the project, the modules were secured to a central post made from stacked PCB pillars, and turned to resemble the steps of a virtually round staircase, see **Figure 3**. 'Elektor Universal Prototyping Board size-1' (UPB-1 a.k.a. Elex-1) was used to construct all modules and give them a uniform look. The boards were labelled and some duplicated to enable other mathematical functions and configurations to be set up.

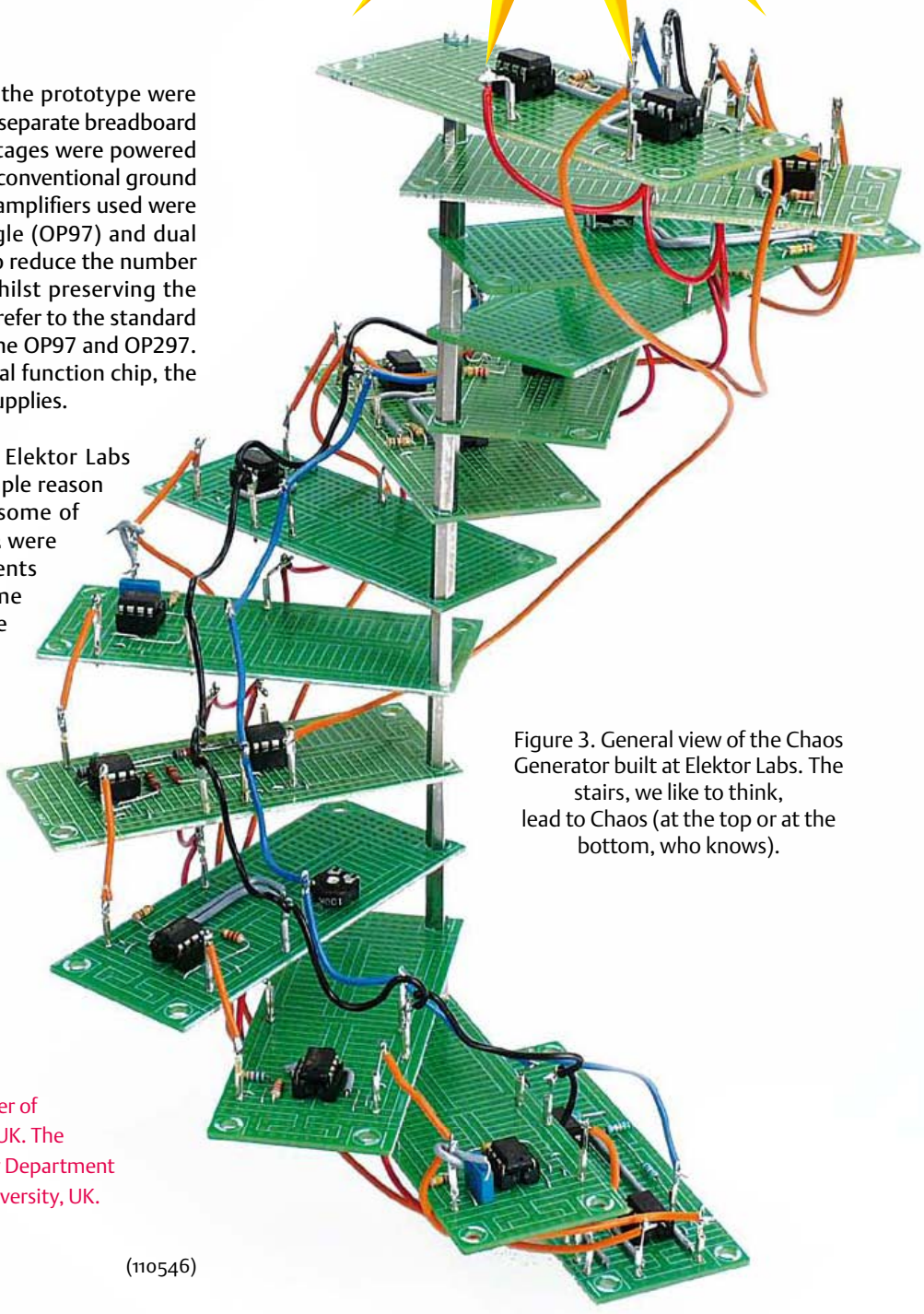


Figure 3. General view of the Chaos Generator built at Elektor Labs. The stairs, we like to think, lead to Chaos (at the top or at the bottom, who knows).

Acknowledgements: This project was stimulated through interdisciplinary workshops of artists and scientists led by artist Charlie Hooker of the University of Brighton Fine Art Department, UK. The analogue computer was built in the Meteorology Department laboratories by Stephen R. Tames, at Reading University, UK.

(110546)

Retronics is a monthly column covering vintage electronics including legendary Elektor designs. Contributions, suggestions and requests are welcomed; please send an email to editor@elektor.com