Conclusions

Morphogenetic processes in nature can be imitated and adapted to control massive swarms of microscopic agents to assemble complex, hierarchically structured systems. Describing these processes by partial differential equations describing masses of infinitesimal particles helps to ensure that they scale up to very large numbers of microscopic agents, which is what will be required for the self-assembly of very complex structures, organized from the microscale up to the macroscale. In this way, we may hope to produce artifacts of a similar sophistication to those in nature.

[1] Associate Professor Emeritus, Department of Electrical Engineering & Computer Science, University of Tennessee.

[2] R. P. Feynman, "There's plenty of room at the bottom," *Engineering and Science 23*, 1960, 22-36. https://resolver.caltech.edu/CaltechES:23.5.1960Bottom

[3] See, for example, A. Clark, *Being There: Putting Brain, Body, and World Together Again*, Cambridge, MIT Press, 1997, and G. Lakoff, M. Johnson, *Philosophy in the Flesh: The Embodied Mind and its Challenge to Western Thought*, New York, Basic Books, 1999.

[4] B. J. MacLennan, "The U-machine: A model of generalized computation," *International Journal of Unconventional Computing 6*, 2010, 265-283.

[5] On artificial morphogenesis, see for example B. J. MacLennan, "Morphogenesis as a model for nano communication," *Nano Communication Networks 1*, 2010, 199-208, and B. J. MacLennan, "The morphogenetic path to programmable matter," *Proceedings of the IEEE 103*, 2015, 1226-1232.

[6] On morphogenetic engineering, see for example R. Doursat, H. Sayama, O. Michel, "A review of morphogenetic engineering," *Natural Computing 12*, 2013, 517-535, and H. Oh, A. R. Shirazi, C. Sun, Y. Jin, "Bio-inspired self-organising multi-robot pattern formation: A review," *Robotics and Autonomous Systems 91*, 2017, 83-100.

[7] B. J. MacLennan, A. C. McBride, "Swarm intelligence for morphogenetic engineering," in A. Schumann (ed.), *Swarm Intelligence: From Social Bacteria to Human Beings*, Boca Raton, Taylor & Francis / CRC, 2020, 9-54.

[8] B. J. MacLennan, "A morphogenetic program for path formation by continuous flocking," *International Journal of Unconventional Computing 14*, 2019, 91-119.

[9] B. J. MacLennan, "Coordinating swarms of microscopic agents to assemble complex structures," in Y. Tan (ed.), *Swarm Intelligence, Vol. 1: Principles, Current Algorithms and Methods*, PBCE 119, Institution of Engineering and Technology, 2018, Chap. 20, 583-612.

[10] J. Cooke, E. C. Zeeman, "A clock and wavefront model for control of the number of repeated structures during animal morphogenesis," *Journal of Theoretical Biology 58*, 1976, 455-476. M.-L. Dequéant, O. Pourquié, "Segmental patterning of the vertebrate embryonic axis," *Nature Reviews Genetics 9*, 2008, 370-382.

Exploring chaos with analog computers

Bernd Ulmann[1]

This article shows how chaotic systems and their behaviour can be explored using analog computers instead of the now prevalent digital approach.

Many natural systems, even very simple ones such as a damped pendulum with an external driving force exhibit chaotic behaviour. One of the main characteristics of such systems is their extreme sensitivity to changes in initial conditions, something often called the "Butterfly Effect." Although these systems are fully deterministic and thus can be described mathematically in closed form, which implies that their future behaviour is completely determined by their past and thus their initial conditions, they are nonetheless not predictable. The term "chaos" was characterised by Edward Norton Lorenz, one of the founders of modern chaos theory, as follows: "*Chaos: When the present determines the future, but the approximate present does not approximately determine the future.*"[2] Interestingly, Lorenz did his groundbreaking work on a tiny digital computer, a Royal McBee LGP-30[3] which is only marginally suited for exploring chaotic systems at best.

To introduce the idea of an analog computer, a short recapitulation of the basic operation of a stored-program digital computer might help: A modern digital computer (typically) has a fixed internal structure, *i.e.* there are one or more arithmetic logic units (ALU), there is a central memory system (nowadays supplemented by a hierarchy of cache memory subsystems to speed things up), and there is a central control unit in addition to a number of input/output channels, *etc.* All of this is controlled by means of a stream of instructions, an "algorithm," stored in memory. At every moment such a machine executes one or more instructions from memory and may decide which instruction to read and process in the next step based on the result of prior instructions executed. So the execution of a program on such a digital computer is basically strictly sequential. (There are, of course, parallel digital computers but exploring this parallelism and achieving a high degree of parallelism is typically at least difficult and most problems won't scale well with this respect.)

In contrast to this, an analog computer has no fixed internal structure, it even has no memory at all and is not programmed by a sequence of instruction to be executed. At its heart an analog computer consists of a number of computing elements, each of which implements a basic operation such as summation, integration, multiplication, *etc.* Values are typically represented in a (basically) continuous form as voltages or currents and not as sequences of bits. (There are, indeed, "digital analog computers," so-called "Digital Differential Analysers," DDAs for short, but these are outside the scope of this article.) Programming an analog computer means to devise a scheme by which the various computing elements are interconnected in order to form a "model," an "analogue" for the problem to be solved.

Although analog computers have been largely forgotten for the last decades due to the low price and ubiquity of stored program digital computers, they have some advantages over their digital rivals, most notably they are extremely energy efficient (in most modern applications for analog computers this high degree of energy efficiency will be the main driver for their application), they interface well to our analog world, and they are inherently interactive. This interactivity is one of the key advantages when it comes to the study of dynamic systems in general and chaotic systems in special, where a researcher can easily change some parameters and "see" the effect in realtime on some output device such as an oscilloscope.[4]

Figure 1 shows a modern analog computer, an Analog Paradigm Model-1 in its basic configuration. The top chassis contains (among

power supplies) four comparators which can be used to model discontinuities, eight manual precision potentiometers which can be used to set coefficients and initial conditions for a simulation, and a manual control unit which allows the machine to operate either in a manual mode where its operation is controlled by a human operator, or in repetitive-mode where one simulation run is repeated over and over again at (typically) high speed in order to get a flicker free picture on an oscilloscope screen. The lower chassis contains eight multipliers, eight summers, and four integrators. (This is part of the real "magic" of an analog computer – integration is one of its basic functions!)

Fig. 1. Small scale analog computer.

Mathematically, dynamic systems can be described by so-called differential equations (DEQ for short), *i.e.* equations in which the unknown is not a value (this would be typically quite simple to solve) but instead a time-varying function. Such DEQs are the tool of choice when it comes to modelling natural systems and are notoriously hard to solve, in fact most differential equations have no analytical solution, *i.e.* there is no closed mathematical expression describing the solution to such an equation. Thus the only viable way to gain an understanding of the underlying system is by means of simulation, be it digital or analog.

Figure 2 shows part of a typical analog computer setup. What looks like an intricate maze of wires is the actual program (at least part of it as the overall program spans many more computing elements) which directly resembles the mathematical problem being solved. This is in contrast to a stored-program digital computer where the underlying mathematical problem has to be translated into a suitable algorithm yielding the desired solution. This additional and often rather error prone and convoluted step can be skipped altogether with an analog computer, the mathematical formulation of a problem is basically sufficient to program an analog computer (sans scaling, which is out of scope here as well).

One of the simplest systems exhibiting chaotic behaviour is a damped pendulum such as a swing being exhibited to an external driving force that might be of a harmonic nature in the most general case. The behaviour of this oscillatory system can then be represented by a "phase space plot," *i.e.* a graphical description of the time-dependent variation of two or more variables involved in the problem. In the case of a pendulum, its angle, the angular velocity, and the angular acceleration could be of interest. Typically, the angular velocity and acceleration are used as x- and y-coordinates for the phase-space plot describing

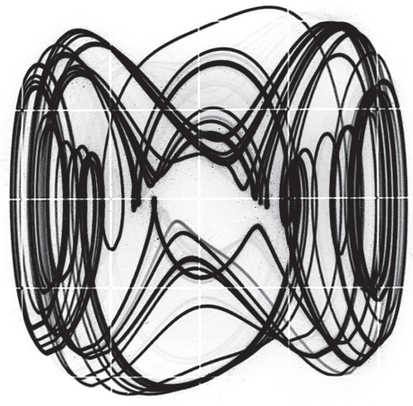

Fig. 3. Phase space plot of a damped pendulum exhibited to an external force.

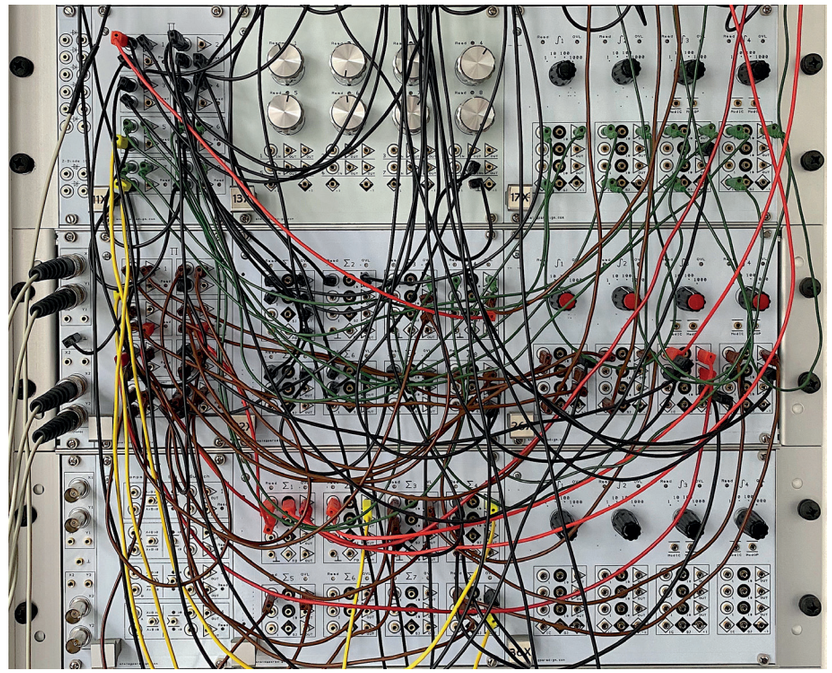

Fig. 2. Typical analog computer setup (partial) – shown are some computing elements such as quad-integrators (on the right side), manual potentiometers (upper middle), multipliers (left)

the behaviour of the system, as shown in the following figure. The chaotic nature of the pendulum's movement can be easily seen.

A figure like in figure 3 is called an "attractor", which is basically a subset of a phase space, *i.e.* a set of values a dynamic system will finally evolve to.

Performing a simulation like this on an analog computer has the advantage that changes in parameters such as the damping or the excitation frequency, *etc.*, can be performed manually by setting potentiometers during run time. This allows one to gain a real "feel" of the behaviour of complex systems as parameter changes take immediate effect and can be directly observed on an oscilloscope.

The next example is the already mentioned original chaotic system first described by Lorenz, a result of early research in atmospheric sciences. He discovered the chaotic nature of this problem through simulations on a digital computer, which was slow and tedious back then. (It is not clear as of now why he did not use an analog computer for his research as the advantages would have been even more profound in his days than they are now.) The system treated by Lorenz is described by three coupled differential equations and represents a simplified model for atmospheric convection.

The three equations involved are parameterised by three coefficients. The original parameter set yields the attractor shown in figure 4.

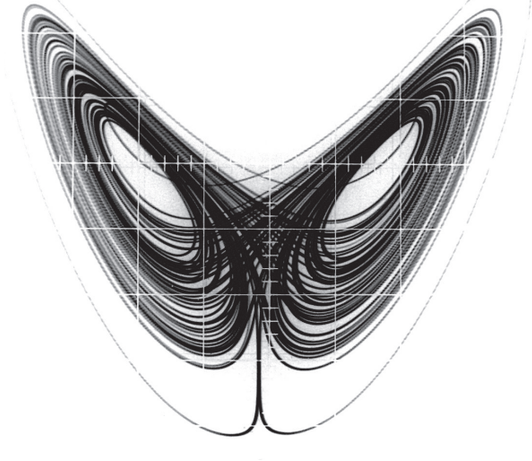

Fig. 4. Typical display of the Lorenz attractor.

This so-called "Lorenz attractor" is a special form of an attractor as it is a "strange attractor," *i.e.* it exhibits a fractal structure. The state of the system at every point in time can be characterised by its corresponding point on this (rather beautiful) graph. The interesting thing is that even tiniest deviations in the initial conditions will lead the system to points far apart on the attractor, so it is not possible to predict the future behaviour of the system. On the other hand, the system cannot "leave" its attractor (at least not for its parameters and initial values within suitable intervals), so the attractor faithfully describes the system behaviour without the chance of actually predicting it from measurements or the like.

All in all, electronic analog computers are ideal tools to explore the behaviour of chaotic (and other dynamic) systems and modern developments in this field will lead to highly integrated analog computers, which will form hybrid computers when combined with classic digital computers thus bringing the advantages of analog computing to a rather wide audience.

[1] Professor for business informatics at the "Hochschule für Oekonomie und Management" in Frankfurt/Main, Germany, and a guest professor and lecturer at the Institute of Medical Systems Biology at Ulm University, ulmann@analogparadigm.com. A number of application notes, including the chaotic systems described here can be found at http://analogparadigm.com/documentation.html, and the author is happy to answer your questions on analog computing.

[2] C. M. Danforth, "Chaos in an Atmosphere Hanging on a Wall," on line: http://mpe.dimacs.rutgers.edu/2013/03/17/chaos-in-an-atmosphere-hanging-on-a-wall/

[3] E. N. Lorenz, "Deterministic Nonperiodic Flow," *Journal of the the Atmospheric Sciences 20*, 1963, 130-141.

[4] For more background information and the history of analog computers, see B. Ulmann, *Analog Computing*, München, De Gruyter Oldenbourg, 2013.

Are There Traces of Megacomputing in Our Universe

Olga Kosheleva and Vladik Kreinovich[1]

The recent successes of quantum computing encouraged many researchers to search for other unconventional physical phenomena that could potentially speed up computations. Several promising schemes have been proposed that will – hopefully – lead to faster computations in the future. Some of these schemes – similarly to quantum computing – involve using events from the micro-world while others involve using large-scale phenomena. If some civilization used micro-world for computations, this will be difficult for us to notice, but if they use mega-scale effects, maybe we can notice these phenomena? In this paper, we analyze what possible traces such megacomputing can leave – and come up with rather surprising conclusions.

Modern computers are fast. By performing billions of computational steps, we can reasonably well predict tomorrow's weather – and when the prediction is not perfect, the problem is usually not with the computers, but with the fact that we do not have enough weather-related sensors in many geographic areas. Onboard computers allow missiles to fly close to the ground at astronomical speeds without hitting the ground. A recent quarantine enables billions of people to be connected by reasonably reliable video-connection, helping many people continue to work, to study, and even to enjoy (remotely) their favorite operas.

But for many practical applications, computers are still too slow. For example, a large part of the US is threatened by destructive tornadoes, and we still do not have a reliable means to predict where a tornado will be moving. As a result, in tornado-prone areas, alarms sound so often – and usually, with no actual tornado coming – that when the actual deadly tornado comes, people do not react to the warning, do not evacuate – and the consequences may be disastrous. Here, we know the equations that would describe the tornado's dynamics – they are largely the same equations that allow us to predict tomorrow's weather. Experiments have shown that by spending the same computation time (several hours) on a supercomputer, we can predict where a tornado will go in the next 15 minutes – but that is too late. This is just one example, there are many other problems like that. Many of such problems are related to organic chemistry and biochemistry. So it is not surprising that, *e.g.* in our university, the main users of high-performance computers are not – as one may think – computer scientists, but folks from the Department of Chemistry and Biochemistry.

How can we make computers faster? Journalists writing about science often express an optimistic belief that human ingenuity will solve all the problems. We are optimistic too, but with computers, we cannot be to optimistic: we are currently reaching the bounds set by fundamental physics. This bound is very simple to explain. According to modern physics, nothing can travel faster that the speed of light – *i.e.* faster than 300 000 km/sec, or 300 000 000 m/sec. How does this affect computations? A usual laptop of which we are typing this article is about 30 cm in diameter, *i.e.* 0.3 m. This means that for a signal to go from one side of the computer to another, we need 0.3/300 000 000 sec, *i.e.* 0.000 000 001 seconds. This may sound like a very small time, but even on the cheapest 4 GigaHerz computer, the processor can perform 4 operations while a signal is still travelling.

To make computers faster, we need to shrink their processing elements even more – this is why we enter the realm of quantum computing.[2] But there is a limit to this shrinkage. Already, a processing element may consist of a few thousand atoms. We can theoretically