Evolutionary Intelligence Special Issue: new emerging areas manuscript No.
(will be inserted by the editor)

Evolution-in-materio: evolving computation in materials

Julian F. Miller · Simon L. Harding · Gunnar Tufte

Received: date / Accepted: date

Abstract Evolution-in-materio (EIM) is the manipulation of a physical system by computer controlled evolution (CCE). It takes the position that to obtain useful functions from a physical system one needs to apply highly specific physical signals and place the system in a particular physical state. It argues that CCE is an effective methodology for doing this. One of the potential advantages of this is that artificial evolution can potentially exploit physical effects that are either too complex to understand or hitherto unknown. EIM is most commonly used as a methodology for implementing *computation* in physical systems. The method is a hybrid of analogue and classical computation in that it uses classical computers to program physical systems or analogue devices. Thus far EIM has only been attempted in a rather limited set of physical and chemical systems. This review paper examines past work related to EIM and discusses historical underpinnings behind such work. It describes latest developments, gives an analysis of the advantages and disadvantages of such work and the challenges that still remain.

Keywords evolutionary algorithms · material computation · evolvable hardware

Julian F. Miller
Department of Electronics, University of York, Heslington, York, UK, YO10 5DD
Tel.: +44 (0)1904 32 2383
Fax: +44 (0)1904 32 2335
E-mail: julian.miller@york.ac.uk

Simon L. Harding
Department of Electronics, University of York, Heslington, York, UK, YO10 5DD
E-mail: slh@evolutioninmaterio.com

Gunnar Tufte
Department of Computer and Information Science, Norwegian University of Science and Technology, 7491 Trondheim, Norway
Tel: +47 73590356
E-mail: gunnar.tufte@idi.ntnu.no

1 Introduction

Natural evolution can be looked at as a physically instantiated algorithm that exploits the physical properties of materials. It is not principled since there is no designer who constructs living systems from simpler and well-understood components. This way of constructing things is in marked contrast to the way human beings design useful systems. Human designers refine useful building blocks and combine these in highly constrained ways to arrive at solutions to problems. This paper is concerned with attempts to use computer controlled evolution to 'program' useful devices by allowing artificial evolution to directly manipulate a physical system. In this way it is hoped to create novel and useful devices in physical systems whose operational principles are hitherto unknown.

We tentatively suggest that it will not be possible to construct computational systems of equivalent power to living organisms using conventional methods that employ purely symbolic methods. We argue that it will be advantageous for complex software systems of the future to utilize physical effects and use some form of search process akin to natural evolution.

Evolution-in-materio (EIM) is a term that refers to the manipulation of physical systems using computer controlled evolution (CCE) [35–37,53]. It is usually thought of as a kind of unconstrained evolution in which by the application of physical signals various intrinsic properties of a material can be heightened or configured so that a useful computational function is achieved. Yoshihito discussed a closely related concept of "material processors" which he describes as material systems that can process information by using the properties of the material [102]. Zauner describes a closely related term which he calls "informed matter" [103].

The concept of EIM grew out of work that arose in a sub-field of evolutionary computation called evolvable hardware [30,38,76,104] particulary through the work of Adrian Thompson [86,91]. In 1996, Thompson famously demonstrated that unconstrained evolution on a silicon chip called a Field Programmable Gate Array (FPGA) could utilize the physical properties of the chip to solve computational problems [85]. We discuss this and related work in Sect. 9 of this paper.

Interestingly, close inspection of much earlier research publications (see Sect. 3) reveals that similar ideas, albeit without computers, were conceived in the late 1950s (particularly by Gordon Pask [63]). In Sect. 6 we discuss Gordon Pask's 1958 work using ferrous sulphate solutions. Evolving computational solutions in other chemical systems has also been achieved, particularly using reaction–diffusion systems. We discuss this in Sect. 11.

Indeed, it is also noteworthy that two of the pioneers of evolutionary computation Ingo Rechenburg and Hans-Paul Schwefel came to invent the *evolutionary strategy* algorithm through a process of trying to manipulate a physical device [69,75]. The device in question, known as an "Experimentum Crucis" was a series of six planar plates that could be positioned at various angles to one another via adjustable hinges. They wanted to minimize the air flow drag across the plate. Using what later became known as a (1+1) evolution-

ary strategy, they made small random changes to the angles and examined the airflow.

Despite this early pioneering work in the evolution of physical devices, the field of evolutionary computation has largely concerned itself with evolution in simulo, which refers to the evolution of *computer models* of physical systems. The reason for this is simple, evolutionary algorithms generally consider many possible solutions to problems in order to converge on a sufficiently satisfactory optimum. In this situation, including "hardware-in-the-loop" is not often feasible, due to the time taken to manipulate the physical system and make an assessment of a potential solution's quality. Since then, computers and computer control systems have become more powerful and sophisticated, this has allowed some physical devices to be manipulated sufficiently quickly for evolution-in-materio to become possible. We review attempts to cross the "reality gap" between simulation of physical objects to evolving buildable designs in Sect. 8.

Eiben et al. [25] discuss what they term "Embodied Artificial Evolution" (EAE) in which evolutionary operators operate in physical space, that is to say the entire algorithm and products of evolution are embodied. They identify three kinds of artificial evolution. In the first kind the evolutionary algorithm and the products are digital, there is no physical artefact produced. Evolutionary optimization, data modeling, simulations etc. fall in this category. In the second kind, the evolution designs a physical artefact (or at least something that could be realised physically), so that something could be constructed after the evolutionary process has finished. In the third kind, the evolutionary process itself and the products of evolution are physical. Directed evolution, in which human beings have intervened in natural evolutionary processes by mutating, recombining and selecting naturally occurring systems (chemicals, enzymes and bacteria) is an example of EAE [73]. They argue that the development of a variety of technologies have made it possible to implement EAE, these include rapid prototyping and modular robotics.

We argue that there is an important stage between the EAE levels 2 and 3, this is where evolution designs a physical artefact which is assessed in situ (i.e. during evolution). So all the evolved designs are embodied however the evolutionary operators still take place inside a computer. This is where most work in evolution-in-materio sits. We refer to this as "embodiment level 2.5." In some systems, the actual evolutionary algorithm may be built in hardware, however usually this is for reasons of computational efficiency or physical compactness. The evolutionary algorithm though operating in purpose built hardware, is functioning in a similar way to an algorithm running on an off-the-shelf computer. Eiben et al.'s level 3 refers to physical systems that are building themselves, so that artefacts are created and destroyed, mutated and recombined.

In EAE the emphasis is on construction. Physical entities are constructed, pass on their genetic material to offspring which are constructed. Through variation, reproduction and selection a form of physically realised evolution takes place. However, in EIM in most cases, the emphasis is on *computation*

although it does include cases where the aim is to obtain physical or chemical systems with desired properties (these are discussed in Sect. 13).

The purpose of this paper is to discuss evolution-in-materio, work that relates to it, and finally to update the reader on the latest developments. Most of the published work we discuss in this paper lies at EA level 2.5. There is a great breadth of work done at this level, and we discuss work in fields as diverse as electronic circuit design, analogue computing, antenna design, chemical computation, protein and enzyme computing, and vesicle design.

One of the exciting aspects of evolution-in-materio is that it has the potential to *discover* new modalities of operation, new physical variables and interactions that can be useful in solving a variety of problems. Evolved systems also have the possibility of offering insights and understanding of exploitable physical processes (in their most general sense) that are hitherto unknown. Using approaches such as artificial curiosity, we may be able to find the problems that are best solved using evolution in materio even if we do not yet know what the problem, or problem types, are [74].

A strong theme in evolution-in-materio is the evolution of novel forms of computation. It appears to be a methodology that has promise in building new kinds of analogue computing devices. Evolution is used as the programming methodology. If unknown physical properties and interactions are to be utilized in a computer program, it is obvious that the the program cannot be designed in advance. An evolutionary algorithm together with a fitness function rewards those configurations of the system that provide the desired computational response. To some extent, we can see the program itself as "emergent", since it is not known in advance and emerges during the evolutionary process.

In section 4 we discuss briefly classical computation and its relation to physics. This is important as EIM uses physical effects and interactions whereas classical computation is based on the concept of a Turing machine in which physics has at best a minor role.

Evolution-in-materio when used in a computational context is a form of analogue computation. Sect. 5 discusses analogue computation and its theoretical foundations. It is interesting to note that Jon Mills's (see Sect. 7) Kirchoff-Lukasiewicz machine arose from Rubel's theoretical definition of an analogue computer that is itself potentially more powerful than Shannon's definition of a general purpose analogue computer.

In a paper in 2002, Miller and Downing coined the term "evolution-in-materio" and suggested a number of physical systems that look both feasible and promising. One of their suggestions was liquid crystal. This led to the work of Harding et al. which demonstrates that it was possible to evolve configurations of liquid crystal so that a liquid crystal display could perform various kinds of computation. This is described in more detail in Section 10.

We also discuss other materials systems where evolution has been used in Sections 13 and 12. In particular, Sect. 12 discusses work showing how nuclear magnetic resonance can be used to carry out classical computation (as opposed to the more usual quantum computation) and how robust logic gates can be

evolved. Sect. 13 discusses work where an evolutionary algorithm was used to find mixtures of chemicals that had high turbidity.

We review briefly what kinds of materials are most suitable for evolution-in-materio in Sect. 14. We also discuss an EU funded project which aims to extensively investigate what kinds of materials are suitable for evolution-in-materio and also what types of computation problems can be solved.

In Sect, 15 we discuss strengths and weaknesses associated with using materials in the evolutionary loop. The paper ends with conclusions and suggestions for further work.

We begin with an overview of the concept of evolution-in-materio. We have tried to be as general as possible so that this depiction encompasses all the various ways that evolution-in-materio may be achieved.

2 Conceptual overview of evolution-in-materio

The central idea of evolution-in-materio is that the application of some physical signals to a material (configuration variables) can cause it to alter how it responds to an applied physical input signal and how it generates a measurable physical output (see Figure 1). Physical outputs from the material are converted to output data and a numerical fitness score is assigned depending on how close a given function of the output data is to that desired. This fitness is assigned to the member of the population that supplied the configuration instructions. Ideally, the material would be able to be reset before the application of new configuration instructions. This is important as without the ability to reset the material it may retain a memory from past configurations, this could lead to the same configuration having different fitness values depending on the history of interactions with the material. We will discuss these advantages and disadvantages in more detail in section 15. We refer to a material that has been configured to perform computation as a *configurable analogue processor* (CAP). For an alternative depiction of a CAP as a FPMA (field programmable matter array) see [53].

A form of CAP is currently being developed in a European FP7-ICT research project called NASCENCE (Nanoscale Engineering for Novel Computation Using Evolution) [15]. In this project a group of researchers are investigating evolution-in-materio using microlectrode arrays. The microelectrode arrays can be covered with a variety of physical materials (see Sect. 14 for details). In this kind of CAP, the physical inputs are voltages applied to electrodes. The physical output are a number of voltages detected at designated output electrodes. The physical configuration is another set of voltages that are applied to electrodes. An evolutionary algorithm decides which electrodes should receive input, output or configuration voltages. With this arrangement it is possible to attempt to solve many well known computational problems. Let us examine a couple of examples.

Suppose that one wishes to use such a CAP to solve a classification problem. In such problems there are a set of input variables (which we assume here

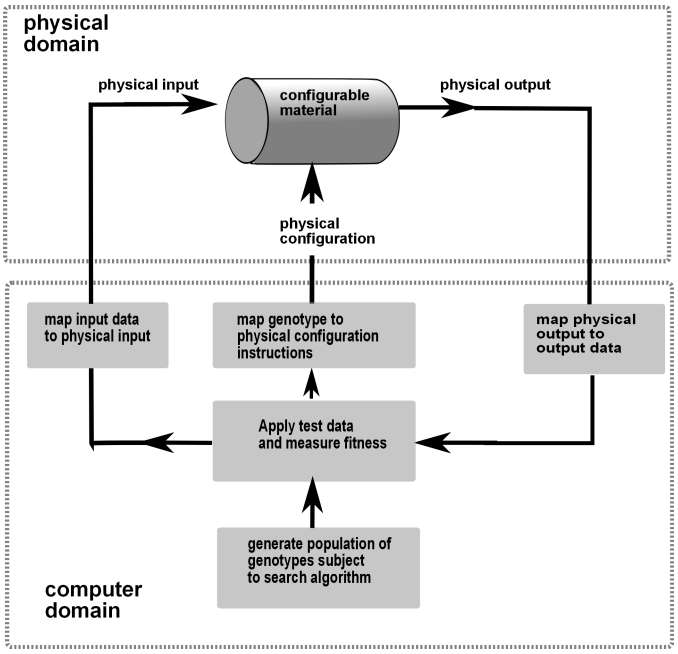


Fig. 1: Concept of evolution-in-materio. There are two domains: physical and computer. In the physical domain there is a material to which physical signals can be applied or measured. These signals are either input signals, output signals or configuration instructions. A computer controls the application of physical inputs applied to the material, the reading of physical signals from the material and the application to the material of other physical inputs known as physical configurations. A genotype of numerical data is held on the computer and is transformed into configuration instructions. The genotypes are subject to an evolutionary algorithm. Physical output signals are read from the material and converted to output data in the computer. A fitness value is obtained from the output data and supplied as a fitness of a genotype to the evolutionary algorithm.

to be real numbers) and a number of classes to which particular instances of input variables belong. On a computer a mapping is created that maps input variables to voltages to be applied to the electrode array. A set of configuration voltages are applied to the electrode array (to electrodes distinct from the electrodes designated as input or output electrodes). The voltages at the output electrodes are measured. On the computer a mapping is created that associates a class with a particular output electrode. The electrode with the largest measured output voltage defines the class of the input instance. The genotype of the evolutionary algorithm is a set of numbers that define config-

uration voltages, the electrodes to which these voltages should be applied and the electrodes which will be used as outputs.

Another computational problem that can be attempted with this approach is the Travelling Salesman problem (TSP). Here one wishes to find the minimum length tour that visits a collection of cities. A solution can be represented by a permutation of the cities. Such a problem is easily mapped onto an electrode array. In this case there are no input electrodes, only output and configuration electrodes. One allocates as many output electrodes as there are cities. The objective is to obtain a set of output voltages which can be mapped into a permutation of cities. This can easily be done as follows. Denote the voltages measured on the output electrodes by V_i. On the computer a two dimensional array a_{ij}, is defined (see Eqn. 1).

$$a_{ij} = \begin{cases} V_i & j = 0 \\ i & j = 1 \end{cases} \tag{1}$$

So a_{i0} holds the measured output voltages and a_{i1} holds the identity permutation. If the array elements a_{i0} are sorted, one can read off a permutation, in a_{i1}. For example in a six-city problem one might record the following output voltages, V_{i0}:

$$\begin{pmatrix} 0.72 & 0.15 & 0.91 & 0.23 & 0.54 & 0.03 \\ 0 & 1 & 2 & 3 & 4 & 5 \end{pmatrix}$$

After sorting by the first dimension we obtain

$$\begin{pmatrix} 0.03 & 0.15 & 0.23 & 0.54 & 0.72 & 0.91 \\ 5 & 1 & 3 & 4 & 0 & 2 \end{pmatrix}$$

It is clear from this discussion that many computational problems can be attempted using an electrode array device. It remains to be seen whether solutions to a variety of computational problems can be *efficiently* evolved using this methodology.

Although "evolution-in-materio" arose relatively recently, it is interesting to note that many of the underlying ideas arose much earlier in the cybernetics community. We review briefly some of these ideas and the main contributors in the next section.

3 Pioneers of ideas related to evolution-in-materio

Gordon Pask's remarkable work in 1958 on electrochemical devices [63] is often cited as one of the pioneering works of evolution-in-materio (We discuss this in Sect. 6). However, Pask's work can be seen as a result of collective information and knowledge gathering that took place in the cybernetic movement.

The cybernetic community emerged mostly in Britain, through the Ratio club [42], and in USA, through the Macy conferences [66], and coincided in time with the development of the concept of stored program computers [61].

However, the cybernetic community also pursued alternative views on information processing in machines (and animals/brains) and pioneered physical computation in terms of philosophy, theory and experimental work. Even before the term cybernetics was coined by Wiener[101] the US cybernetic community was inspired by the largest most complex evolved physical information processing device; the brain. Adaptation and learning in context of feedback loops was the dominant theme. The goals were multitudinous, spanning from understanding neurophysiology to devising self replicating machines, e.g. McCulloch and Pitt's proposal on how networks of neurons could compute logical functions [52] and von Neumann's self-replicating automata [17].

The earliest work may not much resemble modern nanoscale [15] or biology based [3] physical computational devices, but important ideas regarding computational devices, complex systems, emergence and adaptation were sketched out. At the time, adaptation, not evolution, was the main technique suggested for exploring computation (or control).

The British branch of cyberneticians is especially interesting in the context of evolution-in-materio. Adaptation and the brain were the main topics, this was not unexpected since several had a background in medicine or psychiatry [68]. This background led to a view of the brain as a kind of "black box" that, if successfully opened, could be understood. Such a view opened the way for thinking of electromechanical physical computation devices as models for various brain and physiological processes. These early models are connected to the notion of evolution-in-materio in various ways. Like the brain they were trying to explore a potential behaviour (computational capability) which was not known in advance. The atomic units of the system were simple, e.g. a few vacuum tubes, not capable of expressing complex computational behaviour on their own. The behaviour or computation of the systems was a result of several units influencing on each other.

The role and idea of adaptation in the cybernetic community turns out to be quite similar to the idea of using computer controlled evolution in evolution-in-materio. W. Gray Walter's tortoise-like robots [99] were three wheeled robots that could turn the front drive wheel. The controller included only a few vacuum tubes (or two neurons) and a light sensor. The robot controller was hardwired to search for and approach light sources in its environment. The point in relation to evolution-in-materio does not concern the robots themselves, but rather the richness of behaviour and the unexpected life–like behaviour that was found by search and emerged from what was considered a simple physical system interacting in an environment. The brain and adaptation was also the main interest of W. Ross Ashby, as indicated by his most well-known book "Design for a Brain" [7]. His famous machine; the homoestat, demonstrated in the late 1940s, was an analogue and digital mixed signal machine consisting of several interconnected electromechanical units that could communicate and show collective adaptive behaviour. The homeostat demonstrated adaptation, i.e. retaining and re-establishing stability by trial and error. As with Walter's tortoises, the machine itself is not our main interest. However, one design property of Ashby's homeostat should be

emphasized in the context of evolution-in-materio; a mix of analogue control and digital state representation. The mixed signal approach may be a design choice to quantify the number of system states from the richness of analogue processing [20]. This mix of analogue operation and digital state representation can be recognized in evolution-in-materio, e.g. Evolution in silico (Section 9) and Liquid Crystal (Section 10).

Gordon Pask and Stafford Beer were second generation British cyberneticians [68]. They did not share the strong connection to psychiatry of many of their fellows. Their machines were aimed at exploring computation rather than the simulation of brain and its physiological properties. Beer's approach to a physical computational system was related to operational research. Beer sought, in good cybernetic spirit, to automate control of factories and companies. Beer's approach was, as with Walter and Ashby, based on adaptation. In his approach to achieving automatic production, the factory and its control (management) was viewed as an adaptive system adapting to the demands and resources available. He noted that the complexity of such a task was high, in his words, "exceedingly high".

Beer concluded that great complexity was essentially probabilistic in nature and therefore a probabilistic control system was required. There is not much detail on Beer's approach, but an adaptive biological computer was attempted. Based on ecosystems (in a tank) consisting of fish which had be fed iron particles interacting by electromagnetism and light [9]. Beers's biological computer experiments were not successful, but his reflections on the great complexity of the system and its task suggested that the brain as a "black box" could not usefully be opened and understood in detail [10]. This lack of comprehensibility appears to be shared by many of the evolution-in-materio systems (see in particular Section 9).

Beer also collaborated with Gordon Pask on electrochemical devices [63]. Pask's work is close to evolution-in-materio and described in detail in Section 6. However, in relation to cybernetics the use of trail and error as the search for a computational device should be noted. Pask used a process of trial and error, adjusting voltages in a ferrous-sulphate device in order to grow "dendrites" which was similar to the (1+1) evolutionary strategy devised by Rechenburg and Schwefel's (see Sect. 1). Pask continued the theme of adaptation started by Wiener, Walter and Ashby. However, in contrast to the first generation of cyberneticians Pask did not aim at opening the brain "black box", he wanted to construct computational machines beyond conventional computers. He later named such machines "Maverick Machines" [64].

A project with similarities to Pask's electrochemical device is the Linear Field Trainable (LIFT) experimental electrochemical system by Stewart [79]. Stewart pointed out several properties valid for evolution-in-materio; a system can not be opened and inspected, even a part count may not be possible. A "bulk" production process may be required, similar to a bakery, manufacturing "logic by the pound". The proposed production method is especially interesting. Stewart proposed to stimulate machines in a similar way to the way in which modern programmable chips such as FPGAs are configured. Further,

Stewart proposed that the machines should be a grid of metallic structures where the functionality is defined by learning and adaptation. Stewart's device was based on iron or gold particles in nitric acid placed in high pressure (100 - 2000 psi) containers. LIFT was designed to be a pattern recognition system. In a similar manner to that used by Pask, electrical stimuli were applied in a process of trial and error, gradually stimulating the system toward the goal. LIFT, as with Pask's device, enforces growth of structures. However, the goal of LIFT was to be able to produce several different behaviours, i.e. threshold functions, by learning in reconfigurable matter.

In all the early work presented adaptation and learning provided the exploratory force. However, evolution itself can be considered as adaptation. For example, Holland [39] links biological properties, i.e. growth, control and neurophysiology, with Von Neumann's self-reproducing automata and population based adaptation. Coupling the experimental cybernetic machines with evolution, i.e. evolution-in-materio.

4 Classical computation and physics

One of the main aims of evolution-in-materio is to evolve new kinds of computational devices by exploiting physical interactions in material systems. How does this relate to classical computation?

Classical computation is based on an abstract model of a machine called a Turing Machine [97]. Such a machine can write or erase symbols on a possibly infinite one dimensional tape. Its actions are determined by a table of instructions that determine what the machine will write on the tape (by moving one square left or right) given its state (stored in a state register) and the symbol on the tape. Turing showed that the calculations that could be performed on such a machine accord with the notion of computation in mathematics. The Turing machine is an abstraction and it remains unclear what limitations or extensions to the computational power of Turing's model might be possible using real physical processes. Von Neumann and others at the Institute for Advanced Study at Princeton devised a design for a computer architecture based on the ideas of Turing that formed the foundation of modern stored programs computers. These are digital in operation. Although they are made of physical devices (i.e. transistors), computations are made on the basis of whether a voltage is above or below some threshold. Classical computers are based on a *symbolic* notion of computation.

We can view the Turing machine in the context of evolution-in-materio as discussed in the Section 2. The configurable material is the tape. The physical input is the writing of symbols on the tape. The physical output is the reading of symbols on the tape. The tape is a passive storage medium which does not have physical properties that can be exploited.

Interestingly, the degree to which the underlying physics affects both the abstract notion of computation and its tractability has been brought to prominence through the discovery of quantum computation. Deutsch pointed out

that Turing machines implicitly use assumptions based on physics [24]. Other computational systems utilize other aspects of the physical world. Reaction-diffusion systems utilize the chemistry of excitable media [4] (see Sect. 11). DNA computing, uses the chemistry of DNA interactions [5] [6]. Synthetic biology builds computational operations inside living cells [100].

It is interesting to note that Toffoli says that "Nothing Makes Sense in Computing Except in the Light of Evolution" [92]. He argues firstly that a necessary but not sufficient condition for a computation to have taken place, is when a novel function is produced from a fixed and finite repertoire of components (i.e. logic gates, protein molecules). He suggests that a sufficient condition requires *intention*. That is to say, we cannot argue that computation has taken place unless a system has arisen for a higher purpose (this is why he insists on intention as being a prerequisite for computation). Otherwise, almost everything is carrying out some form of computation (which is not a helpful point of view). Thus a Turing machine does not carry out computations unless it has been programmed to do so, and since natural evolution constructs organisms whose genes have an increased chance of reproduction [23] (the higher 'purpose') we can regard them as carrying out computations. It is in this sense that Toffoli points to the fundamental role of evolution in the definition of a computation as it has provided animals with the ability to have intention.

Conrad criticizes traditional models of computing and he identifies that they have a common feature: at most very limited context dependency [22]. The logic gates in conventional machines implement formally defined input-output behaviors that are independent of the overall design of the circuit. Turing machines read and write symbols on individual tape squares in a manner that does not depend on the arrangement of symbols on the tape. Whereas computation in physical systems (particularly biological) are almost always context dependent.

This brings us to one of the important questions in computation. How can we program a physical system to perform a particular computation? The usual answer has been to construct logic gates and from these build a von Neumann machine (i.e. a digital computer). The mechanism that has been used to devise a computer program to carry out a particular computation is the familiar top-down design process, where ultimately the computation is represented using Boolean operations. According to Conrad this process leads us to pay "The Price of Programmability" [21], whereby in conventional programming and design we proceed by excluding many of the processes that may lead to us solving the problem at hand. Indeed as Zauner notes, programmability is not a strict requirement for information processing systems since the enviable computing capabilities of cells and organisms are not implemented by programs. Artificial neural networks are also not programmable in the normal sense [103].

In this paper we suggest that some form of computer–controlled evolution ought to be an appropriate methodology for arriving at *physical* systems that compute. Since EIM utilizes physical effects to perform computation one can consider it as a methodology for performing analogue computation. In particular, it is a general-purpose method for obtaining *specific* analogue computa-

tional devices. In the next section we examine the theoretical underpinnings of analogue computation. This will help in understanding how EIM relates to analogue computation.

5 Analogue computation

Prior to the invention of digital computers there existed a variety of analogue computing machines. Some of these were purely mechanical (e.g. an abacus, a slide-rule, Charles Babbage's difference engine, Vannevar Bush's Differential Analyser) but later computing machines were built using operational amplifiers [13,51]. Bush's differential analyzer was a mechanical analogue computer that could solve differential equations [18]. Shannon analyzed Bush's machine and formulated the mathematical theory behind it [77]. In so doing, he defined the General Purpose Analogue Computer (GPAC). This is a computer that operates on continuous variables $x_i = x_i(t)$ each of which is a function of an independent variable (often called time) and whose four operations, f_i are defined as:

1. constant: $f_1 = c$, where c is a real-valued constant
2. addition: $f_2(x_1, x_2) = x_1 + x_2$
3. multiplication: $f_3(a, x_1) = ax_1$
4. integration: $f_4(x_1, x_2) = \int_{t_0}^{t} x_1(z)dx_2(z)$

There are restrictions about how these functions can be connected. The output of each function must either become the input of another function or be a final output of the computation. Shannon showed that GPACs can compute any differential algebraic function. However, there has been more theoretical work extending Shannon's model to a wider class of computable functions [29,28].

Rubel defined another mathematical model of a more powerful analogue computing machine which he called an Extended Analogue Computer (EAC) [72]. The EAC can produce functions of any finite number of real variables. In addition, the EAC "produces solutions of a broad class of (initial-and) boundary value problems for *partial* differential equations". Rubel uses a larger array of "black-box" functions than in the GPAC. His functions included adders, multipliers, substituters, inverters, differentiators, and some set-theoretic operators (for more detailed explanations see [60]). The functions must be arranged in a line in which the outputs from one function are fed into the inputs of the next function, no two outputs can be connected to the same input, and each input must be connected to an output. The EAC also has connectivity constraints that must be obeyed for its mathematical properties to be provable.

MacLennan gives a mathematical foundation to computers called *field computers* [50]. These are computers that can "perform (in parallel) transformations on scalar and vector fields".

Analogue computation has focused on what kinds of mathematical functions can be provably computed using collections of mathematical primitives.

In EIM, the mathematical functions computed by primitive components are not known explicitly (if at all) and EIM leaves computer-controlled evolution the job of finding suitable combinations of computational primitives that assist in solving computational problems. It does not use the traditional methodology of building solutions to computational problems by combining relatively simple components in a principled manner. It works in a very unconventional design space [54] in which unusual (or even unknown) components are combined and tested and new designs are searched for by evolution.

6 Gordon Pask's work with Ferrous sulphate

Some might consider that 'evolution-in-materio' began in 1958 in the work of Gordon Pask who worked on experiments to grow neural structures using electrochemical assemblages[19,63,67]. Gordon Pask's goal was to create a device sensitive to either sound or magnetic fields that could perform some form of signal processing. He likened this to an ear. He realised he needed a system that was rich in structural possibilities, and chose to use a metal–salt solution. Using electric currents, thread like metallic structures could self-assemble in an acidic aqueous metal–salt solution (e.g. ferrous sulphate). Changing the electric currents can alter the structure of these wires and their positions and the behavior of the system can be modified through external influence. Pask used an array of electrodes suspended in a dish containing the metal–salt solution, and by applying current (either transiently or a slowly changing source) was able to build iron wires that responded differently to two different frequencies of sound- 50Hz and 100Hz.

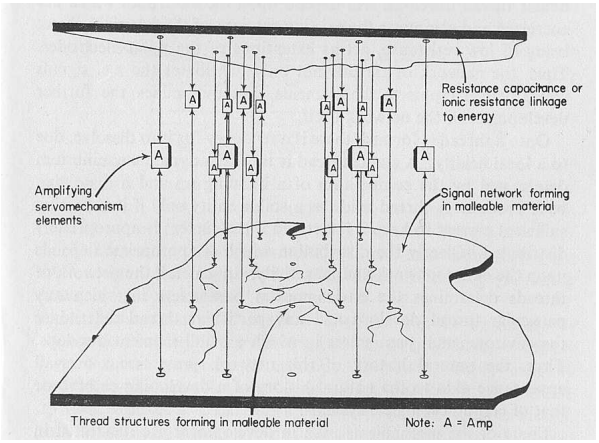


Fig. 2: Pask's experimental set up for growing dendritic wires in ferrous sulphate solution [63]

Pask had developed a system whereby he could manually train the wire formation in such a way that no complete specification had to be given - a complete paradigm shift from previous engineering techniques which would have dictated the position and behavior of every component in the system. His training technique relied on making changes to a set of resistors, and updating the values with given probabilities - in effect a process of generate-and-test. We would today recognise this algorithm as some form of hill climbing algorithm. In a discussion of the epistemological implications of Pask's work, Cariani notes that in Pask's device, the sensory capabilities were emergent, that is to say they were not predefined [19]. To quote:

> Such devices would find their own "relevance criteria", by adaptively constructing sensors to gather the information that they needed to solve a given real world problem. Out of "an infinity of variables" such a device would come up with a set of variables adequate for a specific task. Such a device would be the analogue of both the scientist searching for the right observables for his/her model and the biological evolution of a new sensory modality.

The issue of deciding which physical variables are most useful and appropriate for evolution-in-materio is very difficult. Ideally according to Pask and Cariani, the physical system should "come up" with appropriate variables. However, if one requires a computer-controlled automatic system hard decision have to be taken about what kinds of variables to control. Of course, inside the computational material, other unknown variables may come into play. To some extent these can be seen as internal "relevance criteria". We will discuss this issue further when we consider advantages and disadvantages of evolution-in-materio (see Sect 15).

7 Kirchoff-Lukasiewicz machines: Computing with conductive sheets

Inspired by Rubel's model of an extended analog computer, Mills constructed his own version, which he termed, a Kirchhoff-Lukasiewicz Machine (KLM) [57]. The machines are composed of logical function units connected to a conductive sheet, typically a conductive polymer sheet. The logical units implement Lukasiewicz Logic - a type of multi-valued logic [59, 60]. Figure 3a shows the component parts of an ethernet networked version of the KLM. The Lukasiewicz Logic Arrays (LLA) are connected to the conductive polymer. The LLA bridge areas of the sheet together. The logic units measure the current at one point, perform a transformation and then apply a current source to the other end of the bridge.

Computation is performed by applying current sinks and sources to the conductive polymer and reading the output from the LLAs. Depending on where on the sheet the signals are placed and the configuration of the LLAs, different computations can be performed.

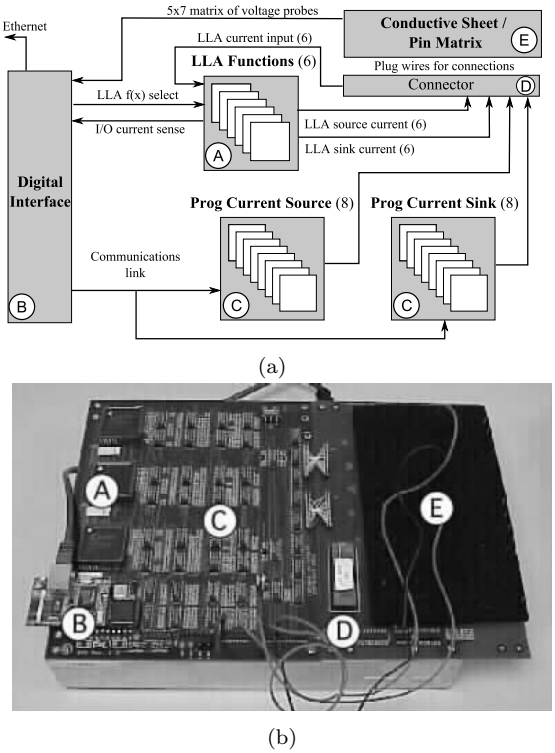


Fig. 3: A labeled schematic of the KLM in 2004 (a) and a photo of a working KLM with labeled components (b).

Examples of computation so far implemented in this system include robot control, control of a cyclotron beam [58], models of biological systems (including neural networks) [56], radiosity based image rendering and a number of other applications [60]. Essentially the KLM performs computation by setting up initial conditions to physically instantiated systems of partial differential equations. Readout happens after these physical effects have stabilized. Using this form of analogue computation, a large number of these equations can be solved in nanoseconds - much faster than on a conventional computer. The speed of computation is dependent on materials used and the interfacing to them, but it is expected that silicon implementations could be capable of solving tens of millions of partial differential equations per second.

The most interesting feature of these devices is the programming method. It is very difficult to understand the actual processes used by the system to perform computation. At the moment, programming requires the manual placement and selection of connections and applied currents. It would be most interesting to investigate the utility of using an evolutionary algorithm to select the functions of the LLAs and to define the current sink/sources. It is interesting to note that although KLMs are clearly using the physical proper-

ties of a material to perform computation, the physical state of the material is not reconfigured.

Mills works suggests that theoretical work on defining types of analogue computation can usefully inform practical work in evolution-in-materio. It highlights the possibility that choosing a conductive sheet as configurable material has real promise. It remains to be seen whether such "simple" materials have advantages over more complex materials such as liquid crystal (see Sect 10).

8 From evolution in simulo to evolved physical artefacts

Partly as a matter of completeness we briefly examine work closely related to evolution-in-materio in which evolution is used to design "buildable objects". In these systems the "configurable material" is not a real material but either a virtual entity or a model of a physical object. Despite not directly manipulating a physical object, it is possible for evolutionary algorithms to find novel solutions to problems that can be subsequently built. Indeed, such systems can show that evolution can utilize hitherto unknown physical effects *in simulation* for problem solving.

Karl Sims created a model of creatures inside a computer [78]. The creatures existed in a virtual world having a simple physics. The creatures were built out of virtual rectangular blocks with several degrees of freedom. The blocks and the creatures were invested with an evolved control system. Sims showed that evolution could arrive at novel designs for creatures that could achieve higher goals (i.e. running or swimming fast). Evolution was able to exploit virtual physical laws in innovative and unexpected ways to arrive at novel solutions.

Funes and Pollack created a simulation of static structures made of Lego [26, 27]. Their evolved simulated novel structures could be physically assembled afterward. The simulation proved to be reasonably faithful so that behaviour of the physical structures was similar to that exhibited in simulation.

Hornby and Pollack investigated evolving Lindemayer systems to define both the neural controllers and the physical assembly of "Tinker-Toy" modular robots which were capable of movement [40].

Lipson and Pollack also simulated the neural control and physical morphology of robots in simulation [47]. However, they carefully defined the simulation of robot morphology so that it could be printed out using a 3D printer. After sufficiently fit robots were evolved and printed, they added motors and connected up the evolved software neural controllers.

Paul et al. evolved simulations of tensegrity robots which could be subsequently built [65]. For a recent review of evolutionary robotics see [14].

Rieffel and Sayles note that few evolved designs are subsequently manufactured into physical objects. One reason for this is that often evolved designs are descriptive rather than being a detailed plan for how the object might be constructed. Another is that for highly complex or flexible physical designs,

there is too large a discrepancy between the simulated properties of the object and its actual physical properties when assembled. In such cases one refers to there being too large a "reality gap". To bridge this gap, Rieffel and Sayles recommend using a device called an "EvoFab" within the evolutionary loop. "EvoFab" is a fully embodied evolutionary fabricator [70].

Bhalla et al. used a rapid prototyping machine to physically realise 3D self-assembling objects whose self-assembly rules had been evolved in a simulator [11].

One of the most successful examples of evolved simulated objects crossing the reality gap into the physical world is Linden's construction of evolved antennas [45].

Segemented wire antennas were evolved in simulation and subsequently assembled. Indeed, an evolved design surpassed a state-of-the-art conventional design and was deployed on a NASA spacecraft in 2006 [49]. Interestingly, to ensure that the evolved antennas would be relatively insensitive to small errors in construction, the team of researchers evaluated the simulation fitness on a number of randomly perturbed designs.

Linden also evolved unusual antennas using reed relays in situ, without requiring a simulator [46]. He assembled various rigid physical structures using reed relays and controlled whether the relays were open or closed via an evolutionary algorithm. He investigated the ability of the antennas to receive transmitted radio waves. This experiment was in effect a form of evolution-in-materio in which the configurable material was a structure made of relays and coils.

Many of these papers illustrate how artificial evolution can exploit simulated physical variables in unexpected and innovative ways. It seems highly likely that utilizing actual physical systems would furnish evolution with a much greater range of exploitable effects. In the next section we discuss how EIM can be easily implemented using programmable arrays of electronic components. Indeed, it was work by researchers in a field known as evolvable hardware that first alerted the evolutionary computation research community about the rich possibilities afforded by allowing evolution to manipulate hardware directly.

9 Evolution in silico

"Intrinsic evolution" is a term often used by researchers in evolvable hardware to mean that electronic circuits are configured by evolved genotypes and the fitness is based on physical measurements of the behaviour of the circuit. The release of open architecture Field Programmable Gate Arrays (FPGA) particularly by Xilinx around 1995, greatly assisted researchers to conduct experiments in intrinisic evolution. FPGAs are electronically reconfigurable circuits. This means that configurations can be downloaded onto an FPGA which define an electronic circuit [30]. A little later, other intrinsic evolution platforms

called Field Programmable Transistor Arrays (FPTAs) were developed [81,43, 95]. These allowed reconfigurable circuits to be built from transistors.

Adrian Thompson started investigating whether it was possible to intrinsically evolve working electronic circuits using the Xilinx FPGA [88,85]. He set himself the task of evolving a digital circuit that could discriminate between an applied 1kHz or 10kHz signal. He found that computer controlled evolution of the configuring bit strings could relatively easily solve this problem. However, when he analyzed the successful circuits he found to his surprise that they worked by utilizing subtle electrical properties of the silicon.

Working with the same FPGA design, Huelsbergen et al. [41] found similar odd behaviour while trying to evolve oscillators. They found that evolved circuits were using transistors in a way that was outside of their normal operating conditions. Thompson also found that if the designs (as defined by the configuration bitstring) were moved to another area of the chip they no longer functioned. They were also very temperature sensitive. He also discovered that parts of the chip, that according to the configuration bitstring, were not involved in the circuit were in fact contributing to the operation of the evolved circuit. Despite painstaking analysis and simulation work he was unable to explain how, or what property was being utilized [89]. This lack of knowledge of how the system works, of course, prevents humans from designing systems that are intended to exploit these subtle and complex physical characteristics. However, it does not prevent exploitation through artificial evolution.

There was concern that such intrinsically evolved circuits would be inherently impractical, so in later work Thompson attempted to evolve circuits under a wide variety of physical conditions in an attempt to obtain designs that were independent of the physical properties of the FPGAs [87,90]. He was able to achieve circuits that were robust to most but not all environmental conditions. Stoica et al. advocated a strategy called "mixtrinsic evolution" in which fitness is evaluated both in hardware and in simulation and found that this increased the likelihood of obtaining portable evolved designs [82]. Through constraints on types of allowed configurations and specially designed evolutionary operators, Trefzer was able to evolve circuits on FPTAs that were portable and whose simulated behaviour accorded with their physical behaviour [96].

Stoica et al. also examined the ability of evolutionary algorithms to reconfigure FPTAs to recover particular functions when the FPTA chip had been damaged by high-energy electrons [80]. Radiation damaged chips lose their functionality. The evolutionary algorithm used controlled the state of about 1,500 switches that determined the circuits configurations on the FTPA programmable device. They found that provided the level of damage was not too great, it was possible for evolution to find new circuit configurations that recovered the original function. Interestingly, often the evolutionary algorithm found a working circuit configuration that *utilized the damaged parts* of the FPTA.

Evolving circuits with an FPGA has limitations as the types of components used are predefined by the manufacturer. Another drawback is that the

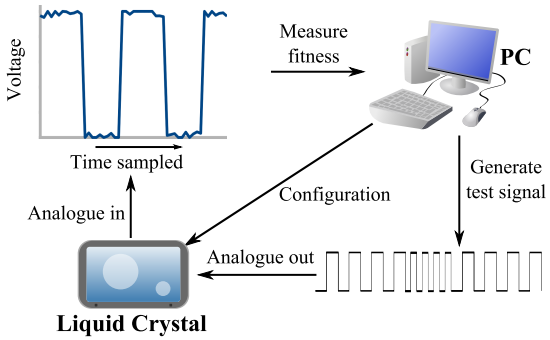
elements of the evolved circuit are not fully testable or easily inspectable. In response to this Paul Layzell created his own platform on which to carry out experiments in intrinsic evolution. He called his circuit an evolvable motherboard (EM) [44]. Layzell performed several experiments with this board to demonstrate that it was a suitable platform for evolution. In the first experiment he evolved a digital inverter (a NOT gate) using two transistors. The evolved design was unconventional, and was found to contain an interesting and unexpected feature - evolution had made no use of the 0V line on the EM. This should have meant that the circuit would not work - however evolution had used an oscilloscope connected for observing the results! The path to 0V required the $10M\Omega$ impedance of the oscilloscopes test lead. Removing the oscilloscope from the circuit stopped the inverter from operating. In another experiment he evolved an oscillator. He found that substituting transistors in evolved functioning solutions with nominally identical ones produced large changes (as much as 30%) in the output frequency - evolution was dependent on the specific electrical characteristics of the components. Some of the evolved circuits exploited features of the environment. The first observation was that some oscillators failed to work if a soldering iron, several meters from the EM, was disconnected from the mains. This was regardless of whether the iron was on or off. The other observation was that some oscillators were 'cheating' and had evolved simple radio receivers that picked up oscillations at the correct frequency [12]. Evolution had used the copper tracks and the other components in the circuit to form aerials, and the signals were connected to the output of the device.

10 Evolved computation in Liquid Crystal

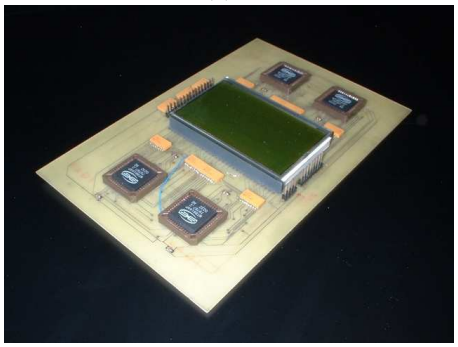
In 2002, Miller and Downing discussed the concept of evolution-in-materio and suggested that liquid crystal might be a suitable material for attempting to evolve computation in materials [53]. They also discussed many other materials whose properties can be affected reversibly via physical signals.

Utilizing the evolvable motherboard concept of Layzell, Harding constructed a liquid crystal analogue processor (LCAP) that utilizes the physical properties of liquid crystal for computation [32]. The experimental setup used by Harding was similar in concept to that used by Thompson [88,85]. This is shown in Fig. 4a(a). A photograph of the LCAP is given in Fig. 4b(b).

Harding used four cross-switch matrix devices to dynamically configure circuits connecting to the liquid crystal. The switches are used to wire the 64 connections on the LCD to one of 8 external connections. The external connections are: input voltages, grounding, signals and connections to measurement devices. Each of the external connectors can be wired to any of the connections on the LCD. The external connections of the LCAP are connected to the I/O card of a PC. One connection was assigned for the incident signal, one for measurement and the other for fixed voltages. The value of the fixed voltages



(a)



(b)

Fig. 4: Liquid Crystal Analogue Processor. (a) Computer-controlled evolutionary loop (b) Photo of device.

is determined by the evolutionary algorithm, but is constant throughout each evaluation. A schematic of the LCAP is given in Fig. 5.

Harding has shown that a number of different computational problems can be evolved using the LCAP:

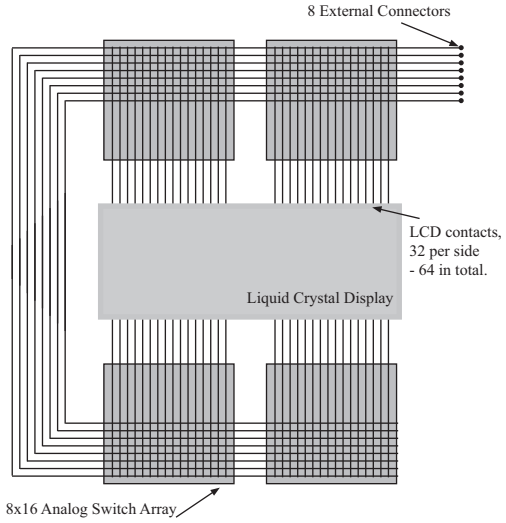


Fig. 5: Schematic of Liquid Crystal Analogue Processor. Part of the genotype defined on the computer decides which electrodes on a LC display will be connected to one of eight contacts on the printed circuit board. Certain contacts are designated to receive square wave signals, others receive fixed analogue voltages and another is connected to ground. The other part of the genotype decides the magnitude of the voltages to apply.

- Tone discriminator. A device was evolved in liquid crystal that could differentiate many different frequencies of square wave [32].
- Logic gates. A variety of two input logic gates were evolved, showing that liquid crystal could behave in a digital fashion. This indicates that liquid crystal is capable of universal computation [36].
- Robot controller. An obstacle avoidance system for a simple exploratory robot was evolved. The results were highly competitive, with other work on evolved robot controllers as it took relatively few evaluations to evolve a working robot controller [33].

It turned out to be relatively easy to evolve the configuration of liquid crystal to solve tasks i.e. only 40 generations of a modest population of configurations are required to evolve a very good frequency discriminator, compared to the thousands of generations required to evolve a similar circuit on an FPGA [88, 85].

11 Reaction-diffusion computers

Reaction diffusion systems are mathematical models which explain how spatially distributed concentrations of chemicals change under the influence of two processes: local chemical reactions and diffusion. Adamatzky pioneered the use of reaction-diffusion systems for computation. He describes reaction-diffusion computers as chemical systems which can process information using interacting growing patterns, of excitable and diffusive chemicals [2]. Data input and output is encoded as concentration profiles of chemical reagents. When different chemical wave fronts interact computation is performed. Such computers are massively parallel since micro–volumes update their states simultaneously, and molecules diffuse and react in parallel. These microvolumes change their state based on local interactions. Inputs and outputs are usually performed optically by supplying illumination masks and for output using imaging systems. Reaction diffusion computational devices have already been shown to solve a range of computational problems, most notably large scale Voronoi diagram construction (which is NP complete). Computationally universal logic gates have been devised. Reaction-diffusion systems can also be controlled via electric fields. Hybrid wetware conventional hardware (webcam and conventional electronics) have been used in "intelligent" robot control [1, 4]

Evolutionary algorithms have been used to control a light-sensitive Belousov-Zhabotinsky (BZ) reaction so that it implements single two–input logic gates (AND, NAND and XOR) [93]. This was achieved by evolving the rules of a heterogeneous cellular automaton to control the light intensity projected onto the surface of a light-sensitive catalyst–infused gel. Also Toth et al. were able to control chemical wave fragments in light-sensitive BZ reactions using evolutionary algorithms to implement collision-based two-input logic gates (AND and NAND) [94]. A learning classifier system XCS has also been used to control chemical-wave fragments in light-sensitive BZ reactions [16].

12 Nuclear magnetic resonance and computation

Nuclear magnetic resonance (NMR) is a physical phenomenon in which nuclei in a magnetic field absorb and then re-emit electromagnetic radiation. Rosello et al. showed how it was possible to implement two-input logic gates by using a variety of radio-frequency pulse parameters. These parameters include pulse amplitude, duration, frequency and phase [71]. In addition, pulses can be applied in sequence.

To implement logic gates one has to decide how to represent binary inputs and how to read the output. One methodology uses pulse sequences which are called pulsed gradient spin echo (PGSE) sequences. In this, one input is decided by whether a starting pulse (a 90 degree pulse) is supplied with a characteristic frequency (resonance) or not. On-resonance represents a 0 and off-resonance represents a 1. The delay before the radio-frequency response of the system is read (data acquisition delay) determines the state of the second

input. If data acquisition (DA) starts immediately after an on-resonance pulse the second input is considered to be 0. If DA starts a given time after the on-resonance pulse, the second input is interpreted as a 1. If the start pulse is off-resonance, and the DA delay is 0, the second input is defined to be 0. Finally, if the start pulse is off-resonance and the DA delay is non-zero, the second input is defined to be 1. One method of deciding whether an output is one or zero is to examine the integral of the spectral intensity (in the frequency domain). A zero total integral is interpreted as an output bit value of 0, a non-zero total integral as 1.

Bechmann et al. pointed out that the underlying continuous spin dynamics can also provide a basis for what they described as "continuous logic operations" [8]. This means that the inputs and outputs are no longer restricted to be the discrete values 0 or 1, but can be any value between 0 and 1. They defined some simple continuous gates (referred to as sin/sin or sin/sinc) and using a form of genetic programming called Cartesian Genetic Programming [55] evolved circuits made of these gates that could function as *digital* (by rounding) even if the inputs have considerable errors (i.e. are real values betwen 0 and 1).

13 Evolution in vitro: optimizing amphiphilic vesicles using genetic algorithms

Thies et al. [84] investigated how a genetic algorithm can optimize, or at least improve, the functionality of an amphiphilic chemical system. Although the aim of their work was not focused on computation it is still interesting to consider as it involves many of the issues that are faced in evolving computation using materials.

Amphiphilic chemicals form vesicles in an appropriate solvent. The genetic algorithm is applied to a population of chemical systems realized in the laboratory. The genetic representation only allowed 5 from 16 amphiphilic chemicals to be chosen. After a time-consuming process of preparation the amphiphiles were mixed in a sucrose solution. Three identical mixtures were prepared for each genotype. The turbidity (or cloudiness) of each of the three identical recipes was measured. Turbidity was chosen merely because it is straightforward to measure, however it correlates well with the size and abundance of vesicles. The fitness of a genotype was defined as the mean turbity minus its standard deviation over the three mixtures. Due to the extreme time consuming nature of the process, only 5 generations using a population of 30 was used. In the evolutionary run they generated a total of 180 recipes which equates to 1.16% of the search space (15,504 possible recipes). It was possible to analyse which recipes acheived the highest fitness and learn about interesting interactions between the chemicals. This is an example of how evolution-in-materio has the potential to generate new science.

14 Materials for evolution-in-materio

It is likely that a search algorithm will to need to examine many thousands of configurations of materials to arrive at set that allows the material to solve a useful computational problem. This implies that ideal materials must be able to be configured in fractions of a second. In addition, it would be helpful to be able to apply test signals and read off responses rapidly. This suggests systems that respond rapidly to electrical signals are most favourable. Miller and Downing discussed a number of potential materials that look promising for evolution-in-materio [53]. These are: liquid crystal, nanoparticle suspensions, molecular films, conducting polymer composites, voltage-controlled colloids, electrorheological fluids and irradiated or otherwise damaged semiconductors. The latter is suggested because it may have more exploitable physics than undamaged silicon. Oltean suggest that electrochromic materials could be used [62].

Rietman showed that it was in principle possible to use a genetic algorithm to program magnetic quantum dots for use in pattern recognition and evolve crystalline programmable resistor arrays for computation [37]. Other promising systems include: Biological or chemical systems that can be manipulated electronically using microfluidics [83] or microbial consortia manipulation using dielectrophoresis [98].

In European FP7-ICT funded NASCENCE project a group of researchers are investigating evolution-in-materio using microlectrode arrays [15]. In this project a variety of materials are being actively investigated for their usefulness in an evolution-in-materio context. Materials such as carbon nanotubes and metallic nanoparticles, can be placed in small chambers surrounded by many tiny electrodes. Using an evolutionary algorithm and computer control, voltages can be applied to electrodes to configure materials to carry out computational tasks. With this device it will be possible to investigate many materials to see if they can be exploited for computation.

It is worth recalling that in Mill's KLM no material is fundamentally changed during configuration. In other words the material is passive, the configuration voltages or currents just determine the initial conditions for sets of physically instantiated partial differential equations. This suggests that it may not be necessary to use complex materials in evolution-in-materio. Further work is required to establish the most suitable material substrates for various kinds of computational problems.

15 Fundamental aspects of evolution-in-materio

We have examined many systems in which computer controlled evolution could be used to configure physical systems so that useful computation could be obtained. Here we discuss possible challenges, advantages and disadvantages of this approach.

Possible advantages are listed below:

1. A computer model of the physical system is not required;
2. Physical variables can be utilized by evolution without knowing that such variables exist in advance;
3. The physical system already exists and does not require assembly after evolution has finished;
4. Allowing evolution to utilize physical effects may increase evolvability;
5. It is possible that hitherto unknown computational devices could be discovered;
6. Computer controlled evolution can explore potential solutions that humans would find inconceivable.
7. The amount of computation a small piece of material can do may be enormous.

Possible disadvantages include:

1. The time taken to apply a configuration of physical signals may be prohibitive;
2. The boundaries of what constitutes the evolved system may be unclear and poorly defined;
3. It is not always clear what physical variables are most appropriate;
4. Applying test signals (over which fitness may be evaluated) may change the intrinsic properties of the system;
5. The system may exhibit weak reproducibility, in that, under identical conditions at different times it may exhibit different behaviour;
6. Evolved physical devices are not universal in a Turing sense. Each problem requires a unique configuration of a physical device to be found.

The advantage 1 may be particularly useful in situations where either simulators do not currently exist or it would be prohibitively expensive to build and run simulations. However, there are some systems that can be simulated faithfully where the computational cost is not prohibitive. An example of this is Linden and Altshuler's work on evolving antennas [45].

Advantage 2 is one of the principal advantages of manipulating matter via an evolutionary algorithm. By merely requiring that a piece of matter perform a particular computation could allow a host of physical processes to be utilized in novel ways. In a sense this is what natural evolution has done with protein chemistry. Evolution-in-materio does not pay "the price of programmability" as the computation is not being performed in the Turing sense.

Advantage 3 is self-evident and is why we argue that evolution-in-materio sits at at level of embodiment equal to 2.5 according to Eiben's scheme. However, although 3 is an advantage, it has the potential disadvantage that it is not clear what the boundaries of the physical system are. More work needs to be done in evolution-in-materio research to study isolation mechanisms. How can we exploit physical effects in an *isolated* system? It is well known how artificial evolution is able to exploit strange unanticipated quirks. Interestingly, this problem also exists in quantum computing, in that the interesting computation happens in very fragile entangled states. To counter Conrad's

phrase one could say that physical computation will have to pay the "price of physicality"!

Potential advantage 4 is intriguing. Harding et al. found that it was relatively easy to evolve a tone discriminator in liquid crystal, much easier than Thompson found it to be in silicon. Conceptually, we can understand this since most physical systems are likely to be endowed with many exploitable properties. We refer to this as *richness*. Silicon based electronic circuits have had the physics constrained as much as possible to implement Boolean logic and thus allow Turing computation. One way of possibly testing the relationship with richness and evolvability would be to attempt to evolve solutions to computational problems on a variety of FPGAs (say) with varying degrees of damage (perhaps by irradiating them). Experiments with radiation damaged FPTAs showed that evolutionary algorithms could make use of damaged parts of the chip (See Sect. 9). It remains to be seen if such damage actually helps solve certain computational problems. Richness and Turing computation appear to be in opposition to one another as Turing computation is based on perfect symbolic operations in which physical effects don't even exist.

Potential advantage 5 is the raison d'etre of evolution-in-materio. Pask, Thompson, Harding and Linden have shown the way. This is why this area of research is exciting, new physical principles could be discovered that allow us to develop new technology. Many computational problems are difficult or intractable to understand. This is a barrier to technological development. The closely related potential advantage 6 suggests that computer-controlled evolution may allow us to develop solutions to problems that are inconceivable to human beings whether they use new physical principles or not. Potential advantage 7 raises many interesting questions. It may be possible to evolve very sophisticated computational processors using a very tiny sample of material. In the NASCENCE project, tiny electrode arrays are being used. How small can we go? This is an intriguing question. It is worth noting that Seth Lloyd has calculated the potential amount of computation possible in matter [48]. He calculated that 1Kg of matter should be able to carry out about 5.510^{50} operations per second and store 10^{31} bits.

Potential disadvantage 1 concerns the time taken to physically configure a material. In many cases, this is a very real problem. Thies et al. described how it took hours to prepare their mixtures of chemicals so that a fitness measurement was possible [84]. Likewise, at present it takes a long time to prepare reaction-diffusion systems (see Sect.11). However in other forms of evolution-in-materio and physical computing systems this is less of a problem (e.g. silicon, liquid crystal and conductive sheets). However, from a programming point of view, it can be considered to take a long time for evolution to configure a system to perform the desired computation. Though this is true, it should be noted that writing conventional computer programs also takes a long time.

Disadvantage 2 is concerned with separability. How can we separate the computation allegedly being carried out by the target device, and that actually done by the material being used. We have already noted examples of

non-separability in Sect. 9. An evolved device may not be useful if it highly sensitive to its environment in unpredictable ways, and it will not always be clear what environmental effects the system is using. To minimise these risks, an evolved system will need to assessed under a range of different environmental conditions. Also as mentioned above methods of isolation need to be investigated.

Disadvantage 3 is a potentially serious problem. Deciding on appropriate physical variables to configure a device is not obvious and normally requires a great deal of insight from a domain expert. To some extent, one should be prepared to investigate a number of ways of configuring materials. Thompson and Harding both used square wave signals as inputs. They both measured the percentage time an output was high as an output signal (i.e. they used digital outputs evaluated over time). Deciding how an input should be applied and an output measured and indeed, what is constitutes an appropriate and useful evolvable physical configuration is a difficult issue. Ideally, we want evolution to construct its own input and output mechanism as in the case of Pask's experiments discussed in Sect. 6. It might be considered that artificial curiosity algorithms could have a role here [74]. However, although curiosity algorithms can build an internal model of behaviour and look for learnable but unknown regularities they can not introduce entirely new physical variables into an existing model. As we observed in 5 the point of evolution-in-materio is that by allowing computer-controlled evolution to apply, read and manipulate physical variables it may be able to exploit possibly unknown modalities of operation. In a very thought-provoking analysis of the epistemological implications of Pask's work Cariani emphasized the importance of techniques that find their own "relevance criteria" [19]. Devising ways of doing computer-controlled evolution so that useful physical variables can be discovered, measured and used for rewarding good partial solutions is a very difficult problem that requires much further research.

Disadvantage 4 was evident in the liquid crystal analogue processor of Harding [31,34]. When the liquid crystal display is observed while solving a problem it is seen that some regions of the liquid display go dark indicating that the local molecular direction has been changed. This means that the configuration of the liquid crystal is changing through the action of test signals. We normally think of such a thing as undesirable. For example, if this happened with conventional computer programs it would mean that the program itself could change in response to test inputs. Thus its behaviour would be dependent on its history. However, viewed from the point of view of a state-machines, this is less worrying. A remedy for this might be merely that we need a reset condition, in which we can place a material in approximately the same initial state. A more pragmatic view would be to say that the fitness function automatically measures the stability of computation over the period of the evaluation. Changes made by the incident signals can be considered part of the genotype-phenotype mapping. Solutions that cannot cope with their initial configurations being altered will achieve a low score. However, the fitness function cannot measure the behaviour beyond the end of the evaluation time.

Therein lies the difficulty, in evolution-in-materio long term stability cannot be guaranteed.

Disadvantage 5 concerns reproducibility. In the case of liquid crystal, liquid crystal molecules are unlikely go back to exactly to a previous state. Stoica et al. also observed that the behaviour of evolved circuits on FPTAs could be dependent of previous configurations[82]. This implies that identical configurations will not produce exactly the same behaviour. This is not necessarily a fundamental problem for evolution provided there is a strong correlation between genotype and phenotype. However, if evolved devices are to be useful in a conventional sense one needs to be sure that previously evolved devices will function in the same way as they did when originally evolved. In Mill's conductive sheets, physical configurations relax back extremely quickly as the substrate is conductive so current dissipates quickly, however there could still be problem if heating occurs as this would change the properties. It is worth noting that natural evolution does not exhibit this exact reproducibility either. Organisms are always different from one another (even if they have the same genome). It is likely that evolved devices will only approximately have the same behaviour.

Disadvantage 6 concerns the potential disadvantages that all analogue computers have. Digital computers are based on the notions of a universal Turing machine. In practical terms this means that digital computers can be programmed to solve many types of computational problems. The data stored in memory changes but the physical architecture does not change. Programs written in high-level languages can be compiled into low level instructions and then executed on the same machine. At present, we are arguing that computer controlled evolution is the method for programming a material. However, using a search process to find a program may not scale to large computational problems. Even if this were true, it may be possible that we can evolve useful functional primitives in materials, which we can later extract and build into analogue "instructions" that more conventional modes of programming can utilize.

16 Conclusions and future outlook

We have discussed attempts to use evolution to manipulate physical systems, particularly but not exclusively, attempts to evolve systems so that they perform useful computation. In our view, computer controlled evolution gives us the possibility to return to analogue computing and build new kinds of computational devices that in the heyday of analogue computing were impossible to imagine. One of the difficulties of analogue computers was that they had to be setup by experts who could create analogues of specific systems in convenient physical devices. Using a search process running on a modern *digital* computer gives us a new way to "program" a physical device. It also gives us a methodology for the discovery of new computational devices that use physical effects for computation in ways that human ingenuity has thus far failed to invent.

The field of evolution-in-materio is still a very young and small community and it is hoped that this paper will serve not only as an introduction to the field but it will encourage more researchers to contribute to this exciting area of research.

References

1. Adamatzky, A.: Computing in Nonlinear Media and Automata Collectives. Institute of Physics Publishing (2001)
2. Adamatzky, A.: Reaction-diffusion computing. In: R.A. Meyers (ed.) Encyclopedia of Complexity and Systems Science, pp. 7548–7565. Springer New York (2009)
3. Adamatzky, A.: Physarum Machines: Computers from Slime Mould. World Scientific Series on Nonlinear Science. World Scientific Publishing Company (2010)
4. Adamatzky, A., Costello, B.D.L., Asai, T.: Reaction-Diffusion Computers. Elsevier (2005)
5. Adleman, L.M.: Molecular computation of solutions to combinatorial problems. Science **266**(11), 1021–1024 (1994)
6. Amos, M.: Theoretical and Experimental DNA Computation. Springer (2005)
7. Ashby, W.R.: Design for a Brain: The Origin of Adaptive Behavior. Chapman & Hall (1960)
8. Bechmann, M., Sebald, A., Stepney, S.: From binary to continuous gates - and back again. In: Proceedings of the 9th international conference on Evolvable systems: from biology to hardware, pp. 335–347 (2010)
9. Beer, S.: A progress note on research into a cybernetic analogue of fabric. In: How Many Grapes Went into the Wine? Stafford Beer on the Art and Science of Holistic Management, Harnden, R. and Allenna, L. (Eds.), Wiley and Sons, reprinted in 1994 (1962)
10. Beer, S.: Towards the automatic factory. In: Transactions of the University of Illinois Symposium on Self-Organization, 1961, pp. 25–89 (1962)
11. Bhalla, N., Bentley, P.J., Vize, P.D., Jacob, C.: Programming and evolving physical self-assembling systems in three dimensions. Natural Computing **11**(3), 475–498 (2012)
12. Bird, J., Layzell, P.: The evolved radio and its implications for modelling the evolution of novel sensors. In: Proceedings of Congress on Evolutionary Computation, pp. 1836–1841 (2002)
13. Bissell, C.: A great disappearing act: the electronic analogue computer. In: IEEE Conference on the History of Electronics, June 28-30. Bletchey, UK (2004)
14. Bongard, J.: Evolutionary robotics. Communications of the ACM **56**(8), 74–85 (2013)
15. Broersma, H., Gomez, F., Miller, J.F., Petty, M., Tufte, G.: Nascence project: Nanoscale engineering for novel computation using evolution. International Journal of Unconventional Computing **8**(4), 313–317 (2012)
16. Bull, L., Budd, A., Stone, C., Uroukov, I., de Lacy Costello, B., Adamatzky, A.: Towards unconventional computing through simulated evolution: Control of nonlinear media by a learning classifier system. Artificial Life **14**, 203–222 (2008)
17. Burks, A.W.: Von neumann's self-reproducing automata. In: A.W. Burks (ed.) Essays On Cellular Automata, pp. 3–64. University of Illinois Press (1970)
18. Bush, V.: The Differential Analyzer: A New Machine for Solving Differential Equations. Journal of the Franklin Institute **212**, 447–488 (1931)
19. Cariani, P.: To evolve an ear: epistemological implications of Gordon Pask's electrochemical devices. Systems Research **3**, 19–33 (1993)
20. Cariani, P.: The homeostat as embodiment of adaptive control. International Journal of General Systems **38**(2), 139–154 (2009)
21. Conrad, M.: The price of programmability. In: R. Herken (ed.) The Universal Turing Machine A Half-Century Survey, pp. 285–307. Oxford University Press (1988)
22. Conrad, M.: Molecular and evolutionary computation: the tug of war between context freedom and context sensitivity. BioSystems **52**, 99–110 (1999)

23. Dawkins, R.: The Selfish Gene. Oxford University Press (1976)
24. Deutsch, D.: Quantum theory, the Church-Turing principle and the universal quantum computer. Proceedings of the Royal Society of London A **400**, 97–117 (1985)
25. Eiben, A.E., Kernbach, S., Haasdijk, E.: Embodied artificial evolution - artificial evolutionary systems in the 21st century. Evolutionary Intelligence **5**(4), 261–272 (2012)
26. Funes, P., Pollack, J.: Computer evolution of buildable objects. In: P. Husbands, I. Harvey (eds.) Fourth European Conference on Artificial Life, pp. 358–367. MIT Press (1997)
27. Funes, P., Pollack, J.: Evolutionary body building: Adaptive physical designs for robots. Artificial Life **4**(4), 337–357 (1998)
28. Graça, D.S.: Some recent developments on Shannon's general purpose analog computer. Mathematical Logic Quarterly **50**(4–5), 473–485 (2004)
29. Graça, D.S., Costa, J.F.: Analog computers and recursive functions over the reals. Journal of Complexity **19**(5), 644–664 (2003)
30. Greenwood, G., Tyrrell, A.M.: Introduction to Evolvable Hardware. IEEE Press (2007)
31. Harding, S.: Evolution in materio. Ph.D. thesis, University of York (2005)
32. Harding, S., Miller, J.F.: Evolution in materio: A tone discriminator in liquid crystal. In: In Proceedings of the Congress on Evolutionary Computation 2004 (CEC'2004), vol. 2, pp. 1800–1807 (2004)
33. Harding, S., Miller, J.F.: Evolution in materio : A real time robot controller in liquid crystal. In: Proceedings of NASA/DoD Conference on Evolvable Hardware, pp. 229–238 (2005)
34. Harding, S., Miller, J.F.: Evolution in materio: Investigating the stability of robot controllers evolved in liquid crystal. In: J.M. Moreno, J. Madrenas, J. Cosp (eds.) Evolvable Systems: From Biology to Hardware, 6th International Conference, Proceedings, *Lecture Notes in Computer Science*, vol. 3637, pp. 155–164. Springer (2005)
35. Harding, S., Miller, J.F.: Evolution in materio. In: R.A. Meyers (ed.) Encyclopedia of Complexity and Systems Science, pp. 3220–3233. Springer (2009)
36. Harding, S.L., Miller, J.F.: Evolution in materio: Evolving logic gates in liquid crystal. International Journal of Unconventional Computing **3**(4), 243–257 (2007)
37. Harding, S.L., Miller, J.F., Rietman, E.A.: Evolution in materio: Exploiting the physics of materials for computation. International Journal of Unconventional Computing **4**(2), 155–194 (2008)
38. Higuchi, T., Liu, Y., Yao, X.: Evolvable hardware. Springer (2006)
39. Holland, J.H.: Outline for a logical theory of adaptive systems. J. ACM **9**(3), 297–314 (1962)
40. Hornby, G., Lipson, H., Pollack, J.B.: Evolution of generative design systems for modular physical robots. In: Proceedings of the 2001 IEEE International Conference on Robotics and Automation, pp. 4146–4151 (2001)
41. Huelsbergen, L., Rietman, E.A., Slous, R.: Evolution of astable multivibrators in silico. In: IEEE Transactions on Evolutionary Computing, vol. 3 (3), pp. 197–204 (1999)
42. Husbands, P., Holland, O.: Mechanical Mind in History, chap. The Ratio Club: A Hub of British Cybernetics, pp. 91–148. MIT Press (2008)
43. Langeheine, J., Becker, J., Folling, S., Meier, K., Schemmel, J.: A CMOS FPTA chip for intrinsic hardware evolution of analog electronic circuits. In: Evolvable Hardware, 2001. Proceedings. The Third NASA/DoD Workshop on, pp. 172–175 (2001)
44. Layzell, P.: A new research tool for intrinsic hardware evolution. Proceedings of The Second International Conference on Evolvable Systems: From Biology to Hardware, LNCS **1478**, 47–56 (1998)
45. Linden, D.S., Altshuler, E.E.: Evolving wire antennas using genetic algorithms: A review. In: 1st NASA / DoD Workshop on Evolvable Hardwaree, pp. 225–232. IEEE Computer Society (1999)
46. Linden, D.S., Altshuler, E.E.: A system for evolving antennas in-situ. In: 3rd NASA / DoD Workshop on Evolvable Hardware, pp. 249–255. IEEE Computer Society (2001)
47. Lipson, H., Pollack, J.B.: Automatic design and manufacture of robotic lifeforms. Nature **406**, 974–978 (2000)
48. Lloyd, S.: Ultimate physical limits to computation. Nature **406**, 1047 – 1054 (2000)

49. Lohn, J.D., Hornby, G.S., Linden, D.S.: Human-competitive evolved antennas. Artificial Intelligence in Engineering Design, Analysis and Manufacturing **22**(3), 235–247 (2008)
50. Maclennan, B.: Field computation: A theoretical framework for massively parallel analog computation. parts i-iv. Tech. Rep. CS-90-100, Department of Computer Science, University of Tennessee (1991)
51. Maclennan, B.: A review of analog computing. Tech. Rep. CS-07-601, Department of Electrical Engineering and Computer Science, University of Tennessee (2007)
52. McCulloch, W., Pitts, W.: A logical calculus of the ideas immanent in nervous activity. The bulletin of mathematical biophysics **5**(4), 115–133 (1943)
53. Miller, J.F., Downing, K.: Evolution in materio: Looking beyond the silicon box. Proceedings of NASA/DoD Evolvable Hardware Workshop pp. 167–176 (2002)
54. Miller, J.F., Job, D., Vassilev, V.K.: Principles in the Evolutionary Design of Digital Circuits – Part I. Genetic Programming and Evolvable Machines **1**(1), 8–35 (2000)
55. Miller, J.F., Thomson, P.: Cartesian genetic programming. In: R. Poli, W.B. et al. (eds.) Proc. of EuroGP 2000, *LNCS*, vol. 1802, pp. 121–132. Springer-Verlag (2000)
56. Mills, J.W.: The continuous retina: Image processing with a single sensor artificial neural field network. Tech. Rep. TR443, Department of Computer Science, University of Indiana (1995)
57. Mills, J.W.: Polymer processors. Tech. Rep. TR580, Department of Computer Science, University of Indiana (1995)
58. Mills, J.W.: Programmable VLSI extended analog computer for cyclotron beam control. Tech. Rep. TR441, Department of Computer Science, University of Indiana (1995)
59. Mills, J.W., Beavers, M.G., Daffinger, C.A.: Lukasiewicz logic arrays. In: Proceedings of 20th International Symposium on Multiple-Valued Logic, pp. 4–10 (1990)
60. Mills, J.W., Parker, M., Himebaugh, B., Shue, C., Kopecky, B., Weilemann, C.: "Empty space" computes: the evolution of an unconventional supercomputer. In: Proceedings of the 3rd conference on Computing frontiers, pp. 115–126. ACM (2006)
61. von Neumann, J.: First Draft of a Report on the EDVAC. Edited by M. D. Godfrey 1992. Technical report, Moore School of Electrical Engineering University of Pennsylvania (1945)
62. Oltean, M.: Switchable glass: A possible medium for evolvable hardware. In: Adaptive Hardware and Systems, NASA/ESA Conference on, pp. 81–87. IEEE Computer Society (2006)
63. Pask, G.: Physical analogues to the growth of a concept. In: Mechanisation of Thought Processes, no. 10 in National Physical Laboratory Symposium, pp. 877–922. Her Majesty's Stationery Office, London, UK (1958)
64. Pask, G., Curran, S.: Micro Man: Computers and the Evolution of Consciousness, chap. 8. Maverick Machines, pp. 133 – 147. Macmillan (1982)
65. Paul, C., Valero-Cuevas, F.J., Lipson, H.: Design and control of tensegrity robots for locomotion. IEEE Transactions on Robotics **22**(5), 944–957 (2006)
66. Pias, C. (ed.): Cybernetics: The Macy-Conferences 1946-1953. Diaphanes, Zürich, Berlin (2003)
67. Pickering, A.: Cybernetics and the Mangle: Ashby, Beer and Pask. Social Studies of Science **32**(3), 413–437 (2002)
68. Pickering, A.: The Cybernetic Brain: Scetches of Another Future. The University of Chicago Press (2010)
69. Rechenberg, I.: Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution. Dissertation. Friedrich Frommann Verlag, 1973 (1971)
70. Rieffel, J., Sayles, D.: EvoFab: a fully embodied evolutionary fabricator. In: Proceedings of the 9th international conference on Evolvable systems: from biology to hardware, pp. 372–380. Springer-Verlag (2010)
71. Roselló-Merino, M., Bechmann, M., Sebald, A., Stepney, S.: Classical computing in nuclear magnetic resonance. International Journal of Unconventional Computing **6**(3-4), 163–195 (2010)
72. Rubel, L.: The extended analog computer. Advances in Applied Mathematics **14**, 39–50 (1993)

73. Rubin-Pitel, S., Cho, C.M.H., Chen, W., Zhao, H.: Bioprocessing for Value-Added Products from Renewable Resources: New Technologies and Applications, chap. Directed Evolution Tools in Bioproduct and Bioprocess Development, pp. 49–72. Elsevier (2006)
74. Schmidüber, J.: Developmental robotics, optimal artificial curiosity, creativity, music, and the fine arts. Connection Science **18**(2), 173–187 (2006)
75. Schwefel, H.P.: Numerische Optimierung von Computer-Modellen. Dissertation. Birkhäuser, Basel, published in 1977 (1974)
76. Sekanina, L.: Evolvable components: From Theory to Hardware Implementations. Natural Computing. Springer (2004)
77. Shannon, C.E.: Mathematical theory of the differential analyzer. Journal of Mathematical Physics MIT **20**, 337–354 (1941)
78. Sims, K.: Evolving virtual creatures. In: Proceedings of the 21st annual conference on Computer graphics and interactive techniques, SIGGRAPH '94, pp. 15–22. ACM (1994)
79. Stewart, R.M.: Electrochemically active field-trainable pattern recognition systems. IEEE Transactions on Systems Science and Cybernetics **5**(3), 230–237 (1969)
80. Stoica, A., Arslan, T., Keymeulen, D., Duong, V., Guo, X., Zebulum, R., Ferguson, I., Daud, T.: Evolutionary recovery of electronic circuits from radiation induced faults. In: Evolutionary Computation, 2004. CEC2004. Congress on, vol. 2, pp. 1786–1793 (2004)
81. Stoica, A., Keymeulen, D., Zebulum, R., Thakoor, A., Daud, T., Klimeck, Y., Tawel, R., Duong, V.: Evolution of analog circuits on field programmable transistor arrays. In: Evolvable Hardware, 2000. Proceedings. The Second NASA/DoD Workshop on, pp. 99–108 (2000)
82. Stoica, A., Zebulum, R.S., Keymeulen, D.: Mixtrinsic evolution. In: Proceedings of the Third International Conference on Evolvable Systems: From Biology to Hardware (ICES2000), *Lecture Notes in Computer Science*, vol. 1801, pp. 208–217. Springer (2000)
83. Tangen, U., Wagler, P.F., Chemnitz, S., Goranovic, G., Maeke, T., McCaskill, J.S.: An electronically controlled microfluidic approach towards artificial cells. Complexus **3**, 48–57 (2006)
84. Theis, M., Gazzola, G., Forlin, M., Poli, I., Hanczyc, M.M., Bedau, M.A.: Optimal formulation of complex chemical systems with a genetic algorithm. In: Online Proceedings of the European Conference on Complex Systems (ECCS '06), p. 193 (2006)
85. Thompson, A.: An evolved circuit, intrinsic in silicon, entwined with physics. In: T. Higuchi, M. Iwata, L. Weixin (eds.) Proc. 1st Int. Conf. on Evolvable Systems (ICES'96), *LNCS*, vol. 1259, pp. 390–405. Springer-Verlag (1997)
86. Thompson, A.: Hardware Evolution: Automatic design of electronic circuits in reconfigurable hardware by artificial evolution. Distinguished dissertation series. Springer-Verlag (1998)
87. Thompson, A.: On the automatic design of robust electronics through artificial evolution. In: M. Sipper, D. Mange, A. Prez-Uribe (eds.) Evolvable Systems: From Biology to Hardware, vol. 1478, pp. 13–24. Springer, New York, NY (1998)
88. Thompson, A., Harvey, I., Husbands, P.: Unconstrained evolution and hard consequences. In: E. Sanchez, M. Tomassini (eds.) Towards Evolvable Hardware: The evolutionary engineering approach, *LNCS*, vol. 1062, pp. 136–165. Springer-Verlag (1996)
89. Thompson, A., Layzell, P.: Analysis of unconventional evolved electronics. Communications of the ACM **42**(4), 71–79 (1999)
90. Thompson, A., Layzell, P.: Evolution of robustness in an electronics design. In: J. Miller, A. Thompson, P. Thomson, T. Fogarty (eds.) Proc. 3rd Int. Conf. on Evolvable Systems (ICES2000): From biology to hardware, *LNCS*, vol. 1801, pp. 218–228 (2000)
91. Thompson, A., Layzell, P., Zebulum, R.S.: Explorations in design space: Unconventional electronics design through artificial evolution. IEEE Transactions on Evolutionary Computation **3**(3), 167–196 (1999)
92. Toffoli, T.: Nothing makes sense in computing except in the light of evolution. International Journal of Unconventional Computing **1**(1), 3–29 (2005)

93. Toth, R., Stone, C., Adamatzky, A., de Lacy Costello, B., Bull, L.: Dynamic control and information processing in the BelousovZhabotinsky reaction using a coevolutionary algorithm. Journal of Chemical Physics **129**, 184708 (2008)
94. Toth, R., Stone, C., de Lacy Costello, B., Adamatzky, A., Bull, L.: Simple Collision-Based Chemical Logic Gates with Adaptive Computing, chap. 11, pp. 162–175. Theoretical and Technological Advancements in Nanotechnology and Molecular Computation: Interdisciplinary Gains. IGI Global (2011)
95. Trefzer, M., Langeheine, J., Meier, K., Schemmel, J.: Operational amplifiers: An example for multi-objective optimization on an analog evolvable hardware platform. In: J. Moreno, J. Madrenas, J. Cosp (eds.) Evolvable Systems: From Biology to Hardware, *Lecture Notes in Computer Science*, vol. 3637, pp. 86–97. Springer (2005)
96. Trefzer, M., Langeheine, J., Schemmel, J., Meier, K.: New genetic operators to facilitate understanding of evolved transistor circuits. In: Evolvable Hardware, 2004. Proceedings. 2004 NASA/DoD Conference on, pp. 217–224 (2004)
97. Turing, A.M.: On computable numbers, with an application to the entscheidungsproblem. Proceedings of the London Mathematical Society **42**(2), 230–265 (1936)
98. Verduzco-Luque, C.E., Alp, B., Stephens, G., Markx, G.: Construction of biofilms with defined internal architecture using dielectrophoresis and flocculation. Biotechnology and Bioengineering **83**(1), 39–44 (2003)
99. Walter, W.G.: The Living Brain. Gerald Duckworth & Co. LTD (1953)
100. Weiss, R., Basu, S., Hooshangi, S., Kalmbach, A., Karig, D., Mehreja, R., Netravali, I.: Genetic circuit building blocks for cellular computation, communications, and signal processing. Natural Computing **2**(1), 47–84 (2003)
101. Wiener, N.: Cybernetics: or Control and Communication in the Animal and the Machine. MIT Press (1948)
102. Yoshihito, A.: Information processing using intelligent materials - information-processing architectures for material processors. Intelligent Materials Systems and Structures **5**, 418–423 (1994)
103. Zauner, K.P.: From prescriptive programming of solid-state devices to orchestrated self-organisation of informed matter. In: J.P. Banâtre, P. Fradet, J.L. Giavitto, O. Michel (eds.) Unconventional Programming Paradigms: International Workshop UPP 2004, vol. 3566, pp. 47–55. Springer (2004)
104. Zebulum, R., Pacheco, M., Vellasco, M.: Evolutionary Electronics – Automatic Design of Electronic Circuits and Systems by Genetic Algorithms. The CRC Press International Series on Computational Intelligence (2002)