# EVOLUTION IN MATERIO

Simon Harding[1] and Julian F. Miller[2]
[1]*Department of Computer Science, Memorial University, NL, Canada*
[2]*Department of Electronics, University of York, UK*

## Article Outline

## Glossary

**Evolutionary algorithm** A computer algorithm loosely inspired by Darwinian evolution

**Generate-and-test** The process of generating a potential solution to a computational problem and testing it to see how good a solution it is. The idea behind it is that no human ingenuity is employed to make good solutions more likely.

**Genotype** A string of information that encodes a potential solution instance of a problem and allows its suitability to be assessed

**Evolution in materio** The method of applying computer controlled evolution to manipulate or configure a physical system

**Liquid Crystal** Substances that have properties between those of a liquid and a crystal

## 1. Definition of the Subject and its Importance

Evolution in materio refers to the use of computers running search algorithms, called evolutionary algorithms, to find the values of variables that should be applied to material systems so that they carry out useful computation. Examples of such variables might be the location and magnitude of voltages that need to be applied to a particular physical system. Evolution in materio is a methodology for programming materials that utilizes physical effects that the human programmer need not be aware of. It is a general methodology for obtaining analogue computation that is specific to the desired problem domain. Although a form of this methodology was hinted at in the work of Gordon Pask in the 1950s it was not convincingly demonstrated until 1996 by Adrian Thompson, who showed that physical properties of a digital chip could be exploited by computer controlled evolution. This article describes the first demonstration that such a method can be used to obtain specific analogue computation in a non-silicon based physical material (liquid crystal). The work is important for a number of reasons. Firstly, it proposes a general method for building analogue computational devices. Secondly it explains how previously unknown physical effects may be utilized to carry out computations. Thirdly, it presents a method that can be used to *discover* useful physical effects that can form the basis of future computational devices.

## 2. Introduction

## Physical Computation

Classical computation is based on a mathematical model of computation based on an abstract (but physically inspired) machine called a Turing Machine [1]. A Turing machine is a machine that 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 (partly because it uses a possibly infinite tape) and to this day it is still not understood 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 based on the ideas of Turing that has formed the foundation of modern computers. Modern computers 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. Prior to the invention of digital computers there have been 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 [2].

There are many aspects of computation that were deliberately ignored by Turing in his model of computation. For instance, speed, programmability, parallelism, openess, adaptivity are not considered. The speed at which an operation can be performed is clearly an important issue since it would be of little use to have a machine that can calculate any computable function but takes an arbitrarily large amount of time to do so. Programmability is another issue that is of great importance. Writing programs directly in the form of instruction tables that could be used with a device based on a Turing is extremely tedious. This is why many high-level computer languages have been devised. The general issue of how to subdivide a computer program into a number of parallel executing processes so that the intended computation is carried out as quickly as possible is still unsolved. Openness refers to systems that can interact with an external environment during their operation. Openness is exhibited strongly in biological systems where new resources can be added or removed either by an external agency or by the actions taken by the system itself. Adaptivity refers to the ability of systems to change their characteristics in response to an environment.

In addition to these aspects, the extent 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, where Deutsch pointed out that Turing machines implicitly use assumptions based on physics [3]. He also showed that through 'quantum parallelism' certain computations could be performed much more quickly than on classical computers. Other forms of physical computation that have recently been explored are:

reaction-diffusion systems [4], DNA computing [5] [6] and synthetic biology [7].

In the UK a number of Grand Challenges in computing research have been proposed [8], in particular 'Journeys in Non-Classical Computation' [9] [10] seeks to explore, unify and generalize many diverse non-classical computational paradigms to produce a mature science of computation.

Toffoli argued that 'Nothing Makes Sense in Computing Except in the Light of Evolution' [11]. He argues firstly that a necessary but not sufficient condition for a computation to have taken place, is when a novel functions 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 that have an increased chance of survival (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.

This brings us to one of the fundamental questions in computation. How can we program a physical system to perform a particular computation? The dominant method used to answer this question 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" [12], whereby in conventional programming and design we proceed by excluding many of the processes that may lead to us solving the problem at hand. Natural evolution does not do this. It is noteworthy that natural evolution has constructed systems of extraordinary sophistication, complexity and computational power. We argue that it is not possible to construct computational systems of such power using a conventional methodology and that complex software systems that directly utilize physical effects will require some form of search process akin to natural evolution together with a way of manipulating the properties of materials. We suggest that some form of evolution ought to be an appropriate methodology for arriving at *physical* systems that compute. In this chapter we discuss work that has adopted this methodology. We call it evolution in materio.

## Evolutionary algorithms

Firstly we propose that to overcome the limitations of a top-down design process, we should use a more unconstrained design technique that is more akin to a process of generate-and-test. However a guided search method is also required that spends more time in areas of the search space that confer favorable traits for computation. One such approach is the use of evolutionary algorithms. These algorithms are inspired by the Darwinian concepts of survival of the fittest and the genetic inheritance of information. Using a computer, a population of randomly generated solutions is systematically tested, selected and modified until a solution has been found [13–15].

As in nature, a genetic algorithm optimizes a population of individuals by selecting the ones that are best suited to solving a problem and allowing their genetic make-up to propagate into future generations. It is typically guided only by the evolutionary process and often contains very limited domain specific knowledge. Although these algorithms are bio-inspired, it is important that any analogies drawn with nature are considered only as analogies.

Their lack of specialization for a problem makes genetic algorithms ideal search techniques where little is known about a problem. As long as a suitable representation is chosen along with a fitness function that allows for ease of movement around a search space, a GA can search vast problem spaces rapidly. Another feature of their behavior is that provided that the genetic representation chosen is sufficiently expressive the algorithm can explore potential solutions that are unconventional. A human designer normally has a set of predefined rules and strategies that they adopt to solve a problem. These preconceptions may prevent trying a new method, and may prevent the designer using a better solution. A genetic algorithm does not necessarily require such domain knowledge. Evolutionary algorithms have been shown to be competitive or surpass human designed solutions in a number of different areas. The largest conference on evolutionary computation called GECCO has an annual session on evolutionary approaches that have produced human competitive scientific and technological results. Moreover the increase in computational power of computers makes such results increasingly more likely.

Many different versions of genetic algorithms exist. Variations in representations and genetic operators change the performance characteristics of the algorithm, and depending on the problem, people employ a variety of modifications of the basic algorithm. However, all the algorithms follow a similar basic set of steps.

Firstly the numbers or physical variables that are required to define a potential solution have to be identified and encoded into a data representation that can be manipulated inside a computer program. This is referred to as the encoding step. The representation chosen is of crucial importance as it is pos-

sible to inadvertedly choose overly constrained representations which limits the portions of the space of potential solutions that will be considered by the evolutionary algorithm. Generally the encoded information is referred to as a genotype and genotypes are sometimes divided into a number of separate strings called chromosomes. Each entry in the chromosome string is an allele, and one or more of these make up a gene.

The second step is to create inside the computer a number of independently generated genotypes whose alleles have been chosen with uniform probability from the allowed set of values. This collection of genotypes is called a population.

In its most basic form, an individual genotype is a single chromosome made of 1s and 0s. However, it is also common to use integer and floating-point numbers if they more appropriate for the task at hand. Combinations of different representations can also be used within the same chromosome, and that is the approach used in the work described in this article. Whatever representation is used, it should be able to adequately describe the individual and provide a mechanism where its characteristics can be transferred to future generations without loss of information.

Each of these individuals is then decoded into its phenotype, the outward, physical manifestation of the individual and tested to see how well the candidate solution solves the problem at hand. This is usually returned as a number that is referred to as the *fitness* of the genotype. Typically it is this phase in a genetic algorithm that is the most time consuming.

The next stage is to select what genetic information will proceed to the next generation. In nature the fitness function and selection are essentially the same - individuals that are better suited to the environment survive to reproduce and pass on their genes. In the genetic algorithm a procedure is applied to determine what information gets to proceed.

Genetic algorithms are often generational - where all the old population is removed before moving to the next generation, in nature this process is much less algorithmic. However, to increase the continuity of information between generations, some versions of the algorithm use elitism, where the fittest individuals are always selected for promotion to the next generation. This ensures that good solutions are not lost from the population, but it may have the side effect of causing the genetic information in the population to converge too quickly so that the search stagnates on a sub-optimal solution.

To generate the next population, a procedure analogous to sexual reproduction occurs. For example, two individuals will be selected and they will then have their genetic information combined together to produce the genotype for the offspring. This process is called recombination or crossover. The genotype is split into sections at randomly selected points called crossover points. A

"simple" GA has only one of these points, however it is possible to perform this operation at multiple points.

Sections of the two chromosomes are then put together to form a new individual. This individual shares some of the characteristics of both parents. There are many different ways to choose which members of the population to breed with each other, the aim in general is to try and ensure that fit individuals get to reproduce with other fit individuals. Individuals can be selected with a probability proportional to their relative fitness or selected through some form of tournament, which may choose two or more chromosomes at random from the population and select the fittest.

In natural recombination, errors occur when the DNA is split and combined together. Also, errors in the DNA of a cell can occur at any time under the influence of a mutagen, such as radiation, a virus or toxic chemical. The genetic algorithm also has mutations. A number of alleles are selected at random and modified in some way. For a binary GA, the bit may be flipped, in a real-numbered GA a random value may be added to or subtracted from the previous allele.

Although GAs often have both mutation and crossover, it is possible to just to use mutation. A mutation only approach has in some cases been demonstrated to work, and often crossover is seen as a macro mutation operator - effectively changing large sections of a chromosome.

After the previous operations have been carried out, the new individuals in the population are then retested and their new fitness scores calculated. Eventually this process leads to an increase in the average fitness of the population, and so the population moves closer toward a solution. This cycle of test, select and reproduce is continued until a solution is found (or some other termination condition is reached), at which point the algorithm stops. The performance of a genetic algorithm is normally measured in terms of the number of evaluations required to find a solution of a given quality.

## Evolution in materio: the historical background

It is arguable that 'evolution in materio' began in 1958 in the work of Gordon Pask who worked on experiments to grow neural structures using electrochemical assemblages[16–19]. Gordon Pask's goal was to create a device sensitive to either sound or magnetic fields that could perform some form of signal processing - a kind of ear. He realised he needed a system that was rich in structural possibilities, and chose to use a metal solution. Using electric currents, wires can be made to 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 - the behavior of the system can be modified through external influence. Pask used an array of electrodes sus-

pended 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.

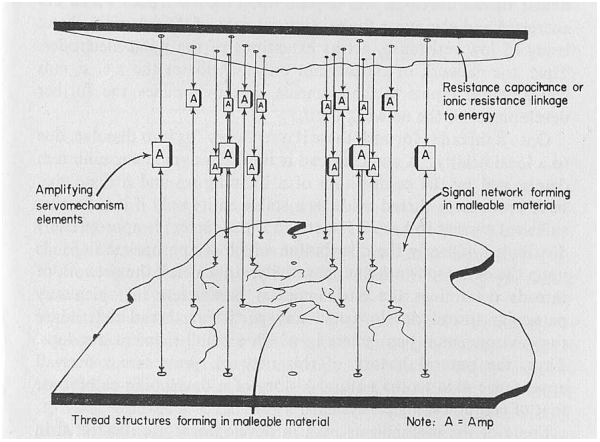


*Figure 1.* Pask's experimental set up for growing dendritic wires in ferrous sulphate solution [17]

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 test-randomly modify-test cycle. We would today recognise this algorithm as some form of evolutionary, hill climbing strategy - with the test stage as the fitness evaluation.

In 1996 Adrian Thompson started what we might call the modern era of evolution in materio. He was investigating whether it was possible to build working electronic circuits using unconstrained evolution (effectively, generate-and-test) using a re-configurable electronic silicon chip called an Field Programmable Gate Array (FPGA). Carrying out evolution by defining configurations of actual hardware components is known as *intrinsic* evolution. This is quite possible using FPGAs which are devices that have a two-dimensional array of logic functions that a configuration bit string defines and connects together. Thompson had set himself the task of evolving a digital circuit that could discriminate between an applied 1kHz or 10kHz applied signal)[20, 21]. 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. Despite painstaking analysis and simulation work he was unable to explain how, or what property was being utilized. 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. Since then a number of researchers have demonstrated the viability of intrinsic evolution in silicon devices [21–27].

The term *evolution in materio* was first coined by Miller and Downing [28]. They argued that the lesson that should be drawn from the work of [21]is that evolution may be used to exploit the properties of a wider range of materials than silicon.

In summary, evolution in materio can be described as:

> Exploitation, using an unconstrained evolutionary algorithm, of the non-linear properties of a malleable or programmable material to perform a desired function by altering its physical or electrical configuration.

Evolution in materio is a subset of a research field known as evolvable hardware. It aims to exploit properties of physical systems with much fewer preconditions and constraints than is usual, and it deliberately tries to avoid paying Conrad's 'Price of Programmability'. However, to get access to physically rich systems, we may have to discard devices designed with human programming in mind. Such devices are often based on abstract idealizations of processes occurring in the physical world. For example, FPGAs are considered as digital, but they are fundamentally analogue devices that have been constrained to behave in certain, human understandable ways. This means that intrinsically complex physical processes are carefully manipulated to represent extremely simple effects (e.g. a rapid switch from one voltage level to another). Unconstrained evolution, as demonstrated by Thompson, allows for the analogue properties of such devices to be effectively utilized.

We would expect physically rich systems to exhibit non-linear properties - they will be complex systems. This is because physical systems generally have huge numbers of parts interacting in complex ways. Arguably, humans have difficulty working with complex systems, and the use of evolution enables us to potentially overcome these limitations when dealing with such systems.

When systems are abstracted, the relationship to the physical world becomes more distant. This is highly convenient for human designers who do not wish to understand, or work with, hidden or subtle properties of materials. Exploitation through evolution reduces the need for abstraction, as it appears evolution is capable of discovering and utilizing any physical effects it can find. The aim of this new methodology in computation is to evolve special purpose computational processors. By directly exploiting physical systems and processes, one should be able to build extremely fast and efficient computational devices. It is our view that computer controlled evolution is a universal methodology

for doing this. Of course, Von Neumann machines (i.e. digital computers) are *individually* universal and this is precisely what confers their great utility in modern technology, however this universality comes at a price. They ignore the rich computational possibilities of materials and try to create operations that are close to a mathematical abstraction. Evolution in materio is a universal methodology for producing specific, highly tuned computational devices.

It is important not to underestimate the real practical difficulties associated with using an unconstrained design process. Firstly the evolved behaviour of the material may be extremely sensitive to the specific properties of the material sample, so each piece would require individual training. Thompson originally experienced this difficulty, however in later work he showed that it was possible to evolve the configuration of FPGAs so that they produced reliable behaviour in a variety of environmental conditions [29].

Secondly, the evolutionary algorithm may utilize physical aspects of any part of the training set-up. Both of these difficulties have already been experienced [21, 23]. A third problem can be thought of as "the wiring problem". The means to supply huge amounts of configuration data to a tiny sample. This problem is a very fundamental one. It suggests that if we wish to exploit the full physical richness of materials we might have to allow the material to grow its own wires and be self-wiring. This has profound implications for intrinsic evolution as artificial hardware evolution requires complete reconfigurability, this implies that one would have to be able to "wipe-clean" the evolved wiring and start again with a new artificial genotype. This might be possible by using nanoparticles that assemble into nanowires. These considerations bring us to an important issue in evolution in materio. Namely, the problem of choosing a suitable materials that can be exploited by computer controlled evolution.

## Evolution in materio: defining suitable materials

The obvious characteristic required by a candidate material is the ability to reconfigure it in some way. Liquid crystal, clay, salt solutions etc can be readily configured either electrically or mechanically; their physical state can be adjusted, and readjusted, by applying a signal or force. In contrast (excluding its electrical properties) the physical properties of an FPGA would remain unchanged during configuration. It is also desirable to bulk configure the system. It would be infeasible to configure every molecule in the material, so the material should support the ability to be reconfigured over large areas using a small amount of configuration.

The material needs to perform some form of transformation (or computation) on incident signals that we apply. To do this, the material will have to interfere with the incident signal and perform a modification to it. We will need to be able to observe this modification, in order to extract the result of

the computation. To perform a non-trivial computation, the material should be capable of performing complex operations upon the signal. Such capabilities would be maximized if the system exhibited non-linear behavior when interacting with input signals.

In summary, we can say that for a material to be useful to evolution in materio it should have the following properties:

- Modify incident signals in observable ways.
- The components of a system (i.e. the molecules within a material) interact with each other locally such that non-linear effects occur at either the local or global levels.
- It is possible to configure the state of the material locally.
- It is possible to observe the state of the material - either as a whole or in one or more locations.
- For practical reasons we can state that the material should be reconfigurable, and that changes in state should be temporary or reversible.

Miller and Downing [28] identified a number of physical systems that have some, if not all, of these desirable properties. They identified liquid crystal as the most promising in this regard as it digitally writable, reconfigurable and works at a molecular level. Most interestingly, it is an example of mesoscopic organization. Some people have argued that it is within such systems that emergent, organized behavior can occur [30]. Liquid crystals also exhibit the phenomenon of self-assembly. They form a class of substances that are being designed and developed in a field of chemistry called Supramolecular Chemistry [31]. This is a new and exciting branch of chemistry that can be characterized as 'the designed chemistry of the intermolecular bond'. Supramolecular chemicals are in a permanent process of being assembled and disassembled. It is interesting to consider that conceptually liquid crystals appear to sit on the 'edge of chaos' [32] in that they are fluids (chaotic) that can be ordered, under certain circumstances.

**Liquid crystal.** Liquid crystal (LC) is commonly defined as a substance that can exist in a mesomorphic state [33, 34]. Mesomorphic states have a degree of molecular order that lies between that of a solid crystal (long-range positional and orientational) and a liquid, gas or amorphous solid (no long-range order). In LC there is long-range orientational order but no long-range positional order.

LC tends to be transparent in the visible and near infrared and quite absorptive in UV. There are three distinct types of LC: lyotropic, polymeric and

thermotropic. Lyotropic LC is obtained when an appropriate amount of material is dissolved in a solvent. Most commonly this is formed by water and amphiphilic molecules: molecules with a hydrophobic part (water insoluble) and a hydrophillic part (strongly interacting with water). Polymeric LC is basically a polymer versions of the aromatic LC discussed. They are characterised by high viscosity and include vinyls and Kevlar. Thermotropic LC (TLC) is the most common form and is widely used. TLC exhibit various liquid crystalline phases as a function of temperature. They can be depicted as rod-like molecules and interact with each other in distinctive ordered structures. TLC exists in three main forms: nematic, cholesteric and smectic. In nematic LC the molecules are positionally arranged randomly but they all share a common alignment axis. Cholesteric LC (or chiral nematic) is like nematic however they have a chiral orientation. In smectic LC there is typically a layered positionally disordered structure. The three types A, B and C are defined as follows. In type A the molecules are oriented in allignment with the natural physical axes (i.e normal to the glass container), however in type C the common molecular axes of orientation is at an angle to the container. LC molecules typically are dipolar. Thus the organisation of the molecular dipoles give another order of symmetry to the LC. Normally the dipoles would be randomly oriented. However in some forms the natural molecular dipoles are aligned with one another. This gives rise to ferroelectric and ferrielectric forms.

There is a vast range of different types of liquid crystal. LC of different types can be mixed. LC can be doped (as in Dye-Doped LC) to alter their light absorption characteristics. Dye-Doped LC film has been made that is optically addressable and can undergo very large changes in refractive index [35]. There are Polymer-Dispersed Liquid Crystals, these can have tailored, electrically controlled light refractive properties. Another interesting form of LC being actively investigated is Discotic LC. These have the form of disordered stacks ( 1-dimensional fluids) of disc-shaped molecules on a two-dimensional lattice. Although discotic LC is an electrical insulator, it can be made to conduct by doping with oxidants [36]. The oxidants are incorporated into the fluid hydrocarbon chain matrix (between disks). LC is widely known as useful in electronic displays, however, there are in fact, many non-display applications too. There are many applications of LC (especially ferroelectric LC) to electrically controlled light modulation: phase modulation, optical correlation, optical interconnects and switches, wavelength filters, optical neural networks. In the latter case a ferroelectric LC is used to encode the weights in a neural network [37].

**Conducting and electroactive polymers.** Conducting polymer composites have been made that rapidly change their microwave reflection coefficient when an electric field is applied. When the field is removed, the composite

reverts to its original state. Experiments have shown that the composite can change from one state to the other in the order of 100ms [38]. Also, some polymers exhibit electrochromism. These substances change their reflectance when a voltage is applied. This can be reversed by a change in voltage polarity [39]. Electroactive polymers [40] are polymers that change their volume with the application of an electric field. They are particularly interesting as voltage controlled artificial muscle. Organic semiconductors also look promising especially when some damage is introduced. Further details of electronic properties of polymers and organic crystals can be found in [41].

**Voltage controlled colloids.** Colloids are suspensions of particles of submicron sizes in a liquid. The phase behavior of colloids is not fully understood. Simple colloids can self assemble into crystals, while multi-component suspensions can exhibit a rich variety of crystalline structures. There are also electrorheological fluids. These are suspensions of extremely fine non-conducting particles in an electrically insulating fluid. The viscosity of these fluids can be changed in a reversible way by large factors in response to an applied electric field in times of the order of milliseconds [42]. Also colloids can also be made in which the particles are charged making them easily manipulatable by suitable applied electric fields. Even if the particles are not charged they may be moved through the action of applied fields using a phenomenon known as dielectrophoresis which is the motion of polarized but electrically uncharged particles in nonuniform electric fields [43]. In work that echoes the methods of Pask nearly four decades ago, dielectrophoresis has been used to grow tiny gold wires through a process of self-assembly [44].

**Langmuir-Blodgett films.** Langmuir-Blodgett films are molecular monolayers of organic material that can be transferred to a solid substrate [45]. They usually consist of hydrophillic heads and hydrophobic tails attached to the substrate. Multiple monolayers can be built and films can be built with very accurate and regular thicknesses. By arranging an electrode layer above the film it seems feasible that the local electronic properties of the layers could be altered. These systems look like feasible systems whose properties might be exploitable through computer controlled evolution of the voltages.

**Kirchoff-Lukasiewicz Machines.** Work by Mills[46, 47] also demonstrates the use of materials in computation. He has designed an 'Extended Analog Computer' (EAC) that is a physical implementation of a Kirchhoff-Lukasiewicz Machine (KLM)[46]. The machines are composed of logical function units connected to a conductive media, typically a conductive polymer sheet. The logical units implement Lukasiewicz Logic - a type of multi-valued logic[47]. Figure 2 shows how 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.

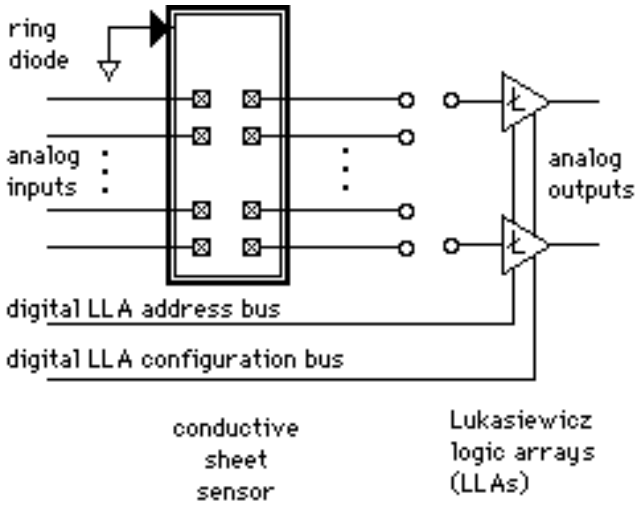


*Figure 2.* Kirchhoff-Lukasiewicz Machine

Computation is performed by applying current sinks and sources to the conductive polymer and reading the output from the LLAs. Different computations can be performed that are determined by the location of applied signals in the conducting sheet and the configuration of the LLAs. Hence, computation is performed by an interaction of the physics described by Kirchoff's laws and the Lukasiewicz Logic units. Together they form a physical device that can solve certain kinds of partial differential equations. 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 how they are interfaced to digital computers, but it is expected that silicon implementations will be capable of finding tens of millions of solutions to the equations per second.

Examples of computation so far implemented in this system include robot control, control of a cyclotron beam [48], models of biological systems (including neural networks) [49] and radiosity based image rendering.

One of 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, and until recently most of the reconfiguration has been done manually. This is difficult as the system is not amenable to traditional software development approaches. However, evolutionary algorithms can be used to automatically define the parameters of the LLAs and the place-

ment of current sinks and sources. By defining a suitable fitness function, the configuration of the EAC can be evolved - which removes the need for human interaction and for knowledge of the underlying system.

Although it is clear that such KLMs are clearly using the physical properties of a material to perform computation, the physical state of the material is not reconfigured i.e. programmed, only the currents in the sheet are changed.

## Evolution in materio is verified with liquid crystal

Harding [50] has verified Miller's intuition about the suitability of liquid crystal as an evolvable material by demonstrating that it is relatively easy to configure liquid crystal to perform various forms of computation.

In 2004, Harding constructed an analogue processor that utilizes the physical properties of liquid crystal for computation. He evolved the configuration of the liquid crystal to discriminate between two square waves of many different frequencies. This demonstrated, for the first time, that the principle of using computer-controlled evolution was a viable and powerful technique for using non-silicon materials for computation. The analogue processor consists of a passive liquid crystal display mounted on a reconfigurable circuit, known as an evolvable motherboard. The motherboard allows signals and configuration voltages to be routed to physical locations in the liquid crystal.

Harding has shown that many different devices can be evolved in liquid crystal including:

- Tone discriminator. A device was evolved in liquid crystal that could differentiate many different frequencies of square wave. The results were competitive, if not superior to those evolved in the FPGA.

- 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.

- Robot controller. An obstacle avoidance system for a simple exploratory robot was evolved. The results were highly competitive, with solutions taking fewer evaluations to find compared to other work on evolved robot controllers.

One of the surprising findings in this work has been that it turns 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. This work has shown that evolving such devices in liquid crystal is easier than when using conventional components, such as FPGAs. The work is a clear demonstration

that evolutionary design can produce solutions that are beyond the scope of human design.

## 3. Evolution In Materio using Liquid Crystal: Implementational details

An evolvable motherboard(EM)[23] is a circuit that can be used to investigate intrinsic evolution. The EM is a reconfigurable circuit that rewires a circuit under computer control. Previous EMs have been used to evolve circuits containing electronic components [23, 51] - however they can also be used to evolve in materio by replacing the standard components with a candidate material.

An EM is connected to an Evolvatron. This is essentially a PC that is used to control the evolutionary processes. The Evolvatron also has digital and analog I/O, and can be used to provide test signals and record the response of the material under evolution.

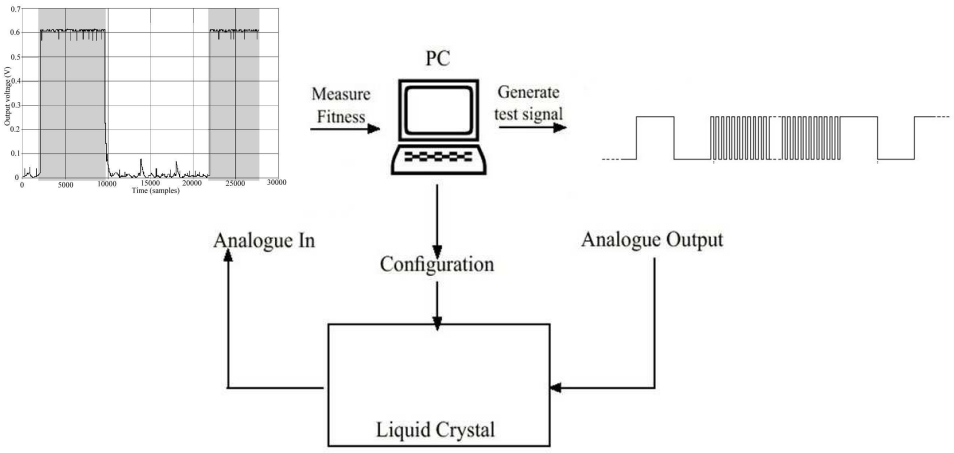


*Figure 3.* Equipment configuration

The Liquid Crystal Evolvable Motherboard (LCEM) is circuit that uses 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 to the LCD.

The external connections of the LCEM are connected to the Evolvatron's analogue inputs and outputs. One connection was assigned for the incident signal, one for measurement and the other for fixed voltages. The value of

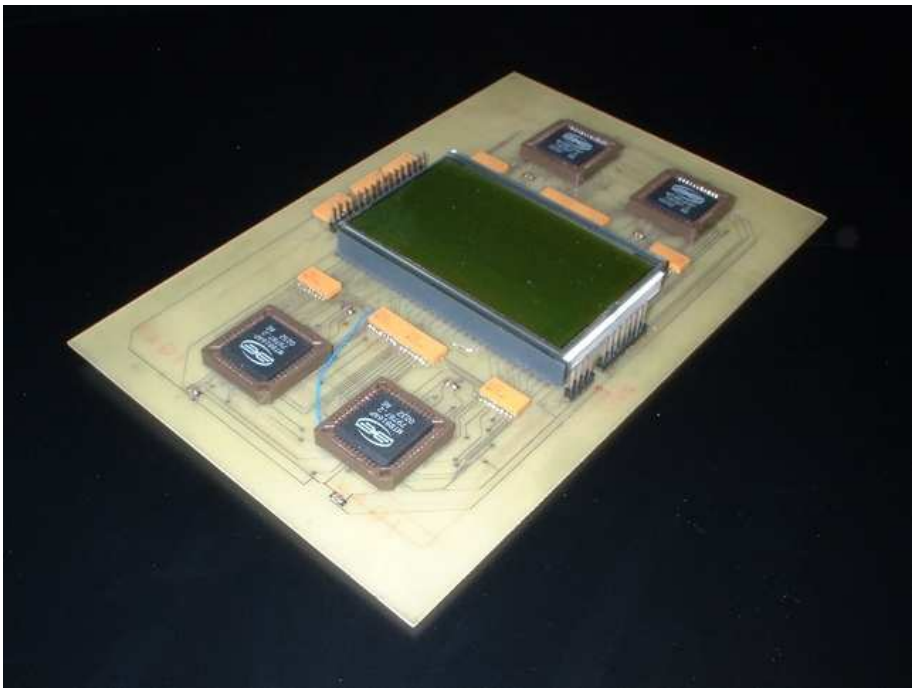

*Figure 4.* The LCEM

the fixed voltages is determined by the evolutionary algorithm, but is constant throughout each evaluation.

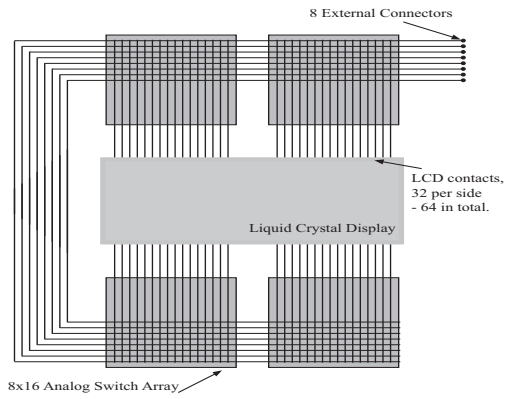


*Figure 5.* Schematic of LCEM

In these experiments the liquid crystal glass sandwich was removed from the display controller it was originally mounted on, and placed on the LCEM. The display has a large number of connections (in excess of 200), however because of PCB manufacturing constraints we are limited in the size of connection we can make, and hence the number of connections. The LCD is therefore roughly positioned over the pads on the PCB, with many of the PCB pads touching more than 1 of the connectors on the LCD. This means that we are applying configuration voltages to several areas of LC at the same time.

Unfortunately neither the internal structure nor the electrical characteristics of the LCD are known. This raises the possibility that a configuration may be applied that would damage the device. The wires inside the LCD are made of an extremely thin material that could easily be burnt out if too much current flows through them. To guard against this, each connection to the LCD is made through a 4.7Kohm resistor in order to provide protection against short circuits and to help limit the current in the LCD. The current supplied to the LCD is limited to 100mA. The software controlling the evolution is also responsible for avoiding configurations that may endanger the device (such as short circuits).

It is important to note that other than the control circuitry for the switch arrays there are no other active components on the motherboard - only analog switches, smoothing capacitors, resistors and the LCD are present.

**Stability and Repeatability Issues.** 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 while signals are being applied. To draw an analogy with circuit design, the incident signals would be changing component values or changing the circuit topology, which would have an effect on the behaviour of the system. This is likely to be detrimental to the measured performance of the circuit. When a solution is evolved, the fitness function automatically measures it stability 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.

Another issue concerns repeatability. When a configuration is applied to the liquid crystal the molecules are unlikely go back to exactly where they were when this configuration was tried previously. Assuming, that there is a strong correlation between genotype and phenotype, then it is likely that evolution will cope with this extra noise. However, if evolved devices are to be useful

one needs to be sure that previously evolved devices will function in the same way as they did when originally evolved.

In [27] it is noted that the behavior of circuits evolved intrinsically can be influenced by previous configurations - therefore their behavior (and hence fitness) is dependent not only on the currently evaluated individuals configuration but on those that came before. It is worth noting that this is precisely what happens in natural evolution. For example, in a circuit capacitors may still hold charge from a previously tested circuit. This charge would then affect the circuits operation, however if the circuit was tested again with no stored charge a different behavior would be expected and a different fitness score would be obtained. Not only does this effect the ability to evolve circuits, but would mean that some circuits are not valid. Without the influence of the previously evaluated circuits the current solution may not function as expected. It is expected that such problems will have analogies in evolution in materio. The configurations are likely to be highly sensitive to initial conditions (i.e. conditions introduced by previous configurations).

**Dealing with Environmental Issues.** A major problem when working with intrinsic evolution is separating out the computation allegedly being carried out by the target device, and that actually done by the material being used. For example, whilst trying to evolve an oscillator Bird and Layzell discovered that evolution was using part of the circuit for a radio antenna, and picking up emissions from the environment [22]. Layzell also found that evolved circuits were sensitive to whether or not a soldering iron was plugged in (not even switched on) in another part of the room![23].

An evolved device is not 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. It would be unfortunate to evolve a device for use in a space craft, only to find out it fails to work once out of range of a local radio tower!

To minimise these risks, we will need to check the operation of evolved systems under different conditions. We will need to test the behavior of a device using a different set up in a different location. It will be important to know if a particular configuration only works with one particular sample of a given material.

## The Computational Power of Materials

In [52] Lloyd, argued that the theoretical computing power of a kilogram of material is far more than is possible with a kilogram of traditional computer. He notes that computers are subject to the laws of physics, and that these laws place limits on the maximum speed they can operate and the amount of information it can process. Lloyd shows that if we were fully able to exploit

a material, we would get an enormous increase in computing power. For example, with 1 kilogram of matter we should be able to perform roughly $5x10^{50}$ operations per second, and store $10^{31}$ bits. Amazingly, contemporary quantum computers do operate near these theoretical limits[52].

A small amount of material also contains a large number of components (regardless of whether we consider the molecular or atomic scale). This leads to some interesting thoughts. If we can exploit materials at this level, we would be able to do a vast amount of computation in a small volume. A small size also hints at low power consumption, as less energy has to be spent to perform an operation. Many components also provides a mechanism for reliability through redundancy. A particularly interesting observation, especially when considered in terms of non Von-Neumann computation, is the massive parallelism we may be able to achieve. The reason that systems such as quantum, DNA and chemical computation can operate so quickly is that many operations are performed at the same time. A programmable material might be capable of performing vast numbers of tasks simultaneously, and therefore provide a computational advantage.

In commercial terms, small is often synonymous with low cost. It may be possible to construct devices using cheaply available materials. Reliability may not be an issue, as the systems could be evolved to be massively fault tolerant using their intrinsic redundancy. Evolution is capable of producing novel designs. Koza has already rediscovered circuits that infringe on recent patents, and his genetic programming method has 'invented' brand new circuit designs [53]. Evolving in materio could produce many novel designs, and indeed given the infancy of programmable materials all designs may be unique and hence patentable.

## 4. Future Directions

The work described here concerning liquid crystal computational devices is at an early stage. We have merely demonstrated that it is possible to evolve configurations of voltages that allow a material to perform desired computations. Any application that ensues from this work is unlikely to be a replacement for a simple electronic circuit. We can design and build those very successfully. What we have difficulty with is building complex, fault tolerant systems for performing complex computation. It appears that nature managed to do this. It used a simple process of a repetitive test and modify, and it did this in a universe of unimaginable physical complexity. If nature can exploit the physical properties of a material and its surroundings through evolution, then so should we.

There are many important issues that remain to be addressed. Although we have made some suggestions about materials worthy of investigation, it is at

present unclear which materials are most suitable. An experimental platform needs to be constructed that allows many materials to be tried and investigated. The use of microelectrode arrays in a small volume container would allow this. This would also have the virtue of allowing internal signals in the materials to be inspected and potentially understood.

We need materials that are rapidly configurable. They must not be fragile and sensitive to minute changes in physical setup. They must be capable of maintaining themselves in a stable configuration. The materials should be complex and allow us to carry out difficult computations more easily than conventional means. One would like materials that can be packaged into small volumes. The materials should be relatively easily interfaced with. So far, material systems have been configured by applying a constant configuration pattern, however this may not be appropriate for all systems. It may be necessary to put the physical system under some form of responsive control, in order to program and then keep the behavior stable.

We may or may not know if a particular material can be used to perform some form of computation. However, we can treat our material as a "black box", and using evolution as a search technique, automatically discover what, if any, computations our black box can perform. The first step is to build an interface that will allow us to communicate with a material. Then we will use evolution to find a configuration we can apply using this platform, and then attempt to find a mapping from a given problem to an input suitable for that material, and a mapping from the materials response to an output. If this is done correctly, we might be automatically able to tell if a material can perform computation, and then classify the computation.

When we evolve in materio, using mappings evolved in software, how can we tell when the material is giving us any real benefit? The lesson of evolution in materio has been that the evolved systems can be very difficult to analyze, and the principal obstacle to the analysis is the problem of separating out the computational role that each component plays in the evolved system. Theses issues are by no means just a problem for evolution in materio. They may be an inherent part of complex evolved systems. Certainly the understanding of biological systems are providing immense challenges to scientists.

The single most important aspect that suggests that evolution in materio has a future is that natural evolution has produced immensely sophisticated material computational systems. It would seem foolish to ignore this and merely try to construct computational devices that operate according to one paradigm of computation (i.e. Turing). Oddly enough, it is precisely the sophistication of the latter that allows us to attempt the former.

## 5. Further Reading

These are just cites placed so that they are included in the bibliography. They will need to be moved to a 2nd bibliography later.

[47, 49, 48, 17, 16, 54] [19, 55–57] [58, 52]

# References


[1] Turing, A.M.: On computable numbers, with an application to the entscheidungsproblem. Proceedings of the London Mathematical Society **42**(2) (1936) 230–265

[2] Bissell, C.: A great disappearing act: the electronic analogue computer. In: IEEE Conference on the History of Electronics, June 28-30. (2004)

[3] Deutsch, D.: Quantum theory, the church-turing principle and the universal quantum computer. Proceedings of the Royal Society of London A **400** (1985) 97–117

[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) (1994) 1021–1024

[6] Amos, M.: Theoretical and Experimental DNA Computation. Springer (2005)

[7] 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) (2003) 47–84

[8] UK Computing Research Committee: Grand challenges in computer research http://www.ukcrc.org.uk/grand$_c hallenges/(2005)$

[9] Stepney, S., Braunstein, S.L., Clark, J.A., Tyrrell, A., Adamatzky, A., Smith, R.E., Addis, T., Johnson, C., Timmis, J., Welch, P., Milner, R., Partridge, D.: Journeys in non-classical computation I: A grand challenge for computing research. International Journal of Parallel, Emergent and Distributed Systems **20**(1) (2005) 5–19

[10] Stepney, S., Braunstein, S., Clark, J., Tyrrell, A., Adamatzky, A., Smith, R., Addis, T., Johnson, C., Timmis, J., Welch, P., Milner, R., Partridge, D.: Journeys in non-classical computation II: Initial journeys and waypoints. International Journal of Parallel, Emergent and Distributed Systems **21**(2) (2006) 97–125

[11] Toffoli, T.: Nothing makes sense in computing except in the light of evolution. International Journal of Unconventional Computing **1**(1) (2005) 3–29

[12] Conrad, M.: The price of programmability. The Universal Turing Machine (1988) 285–307

[13] Goldberg, D.: Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley, Reading, Massachusetts (1989)

[14] Holland, J.: Adaptation in Natural and Artificial Systems. Second edn. MIT Press, Cambridge, Massachusetts (1992)

[15] Mitchell, M.: An introduction to genetic algorithms. MIT Press, Cambridge, MA, USA (1996)

[16] Pask, G.: Physical analogues to the growth of a concept. In: Mechanization of Thought Processes, Symposium 10, National Physical Laboratory. (1958) 765–794

[17] Pask, G.: The natural history of networks. In: Proceedings of International Tracts In Computer Science and Technology and their Application. Volume 2. (1959) 232–263

[18] Cariani, P.: To evolve an ear: epistemological implications of gordon pask's electrochemical devices. In: Systems Research. Volume 3. (1993) 19–33

[19] Pickering, A.: Cybernetics and the mangle: Ashby, beer and pask. Social Studies of Science (2002) 413–437

[20] Thompson, A., Harvey, I., Husbands, P.: Unconstrained evolution and hard consequences. In Sanchez, E., Tomassini, M., eds.: Towards Evolvable Hardware: The evolutionary engineering approach. Volume 1062 of LNCS. Springer-Verlag (1996) 136–165

[21] Thompson, A.: An evolved circuit, intrinsic in silicon, entwined with physics. In: ICES. (1996) 390–405

[22] Bird, J., Layzell, P.: The evolved radio and its implications for modelling the evolution of novel sensors. In: Proceedings of Congress on Evolutionary Computation. (2002) 1836–1841

[23] 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** (1998) 47–56

[24] Linden, D.S., Altshuler, E.E.: A system for evolving antennas in-situ. In: 3rd NASA / DoD Workshop on Evolvable Hardware, IEEE Computer Society (2001) 249–255

[25] Linden, D.S., Altshuler, E.E.: Evolving wire antennas using genetic algorithms: A review. In: 1st NASA / DoD Workshop on Evolvable Hardwaree, IEEE Computer Society (1999) 225–232

[26] Stoica, A., Zebulum, R.S., Guo, X., Keymeulen, D., Ferguson, M.I., Duong, V.: Silicon validation of evolution-designed circuits. In: Proceedings. NASA/DoD Conference on Evolvable Hardware. (2003) 21–25

[27] Stoica, A., Zebulum, R.S., Keymeulen, D.: Mixtrinsic evolution. In: Proceedings of the Third International Conference on Evolvable Systems: From Biology to Hardware (ICES2000). Volume 1801 of Lecture Notes in Computer Science., Springer (2000) 208–217

[28] Miller, J.F., Downing, K.: Evolution in materio: Looking beyond the silicon box. Proceedings of NASA/DoD Evolvable Hardware Workshop (2002) 167–176

[29] Thompson, A.: On the automatic design of robust electronics through artificial evolution. In Sipper, M., Mange, D., Pérez-Uribe, A., eds.: Evolvable Systems: From Biology to Hardware. Volume 1478., Springer, New York, NY (1998) 13–24

[30] Laughlin, R.B., Pines, D., Schmalian, J., Stojkovic, B.P., Wolynes, P.: The middle way. Proceedings of the National Acdemy of Sciences **97**(1) (2000) 32–37

[31] Lindoy, L.F., Atkinson, I.M.: Self-assembly in Supramolecular Systems. Royal Society of Chemistry (2000)

[32] Langton, C.: Computation at the edge of chaos: Phase transitions and emergent computation. In: Emergent Computation. MIT Press (1991) 12–37

[33] Demus, D., Goodby, J.W., Gray, G.W., Spiess, H.W., Vill, V.: Handbook of Liquid Crystals, Handbook of Liquid Crystals: Four Volume Set. Handbook of Liquid Crystals, Handbook of Liquid Crystals: Four Volume Set, by Dietrich Demus (Editor), John W. Goodby (Editor), George W. Gray (Editor), Hans W. Spiess (Editor), Volkmar Vill (Editor), pp. 2180. ISBN 3-527-29502-X. Wiley-VCH , June 1998. (1998)

[34] Khoo, I.C.: Liquid Crystals: physical properties and nonlinear optical phenomena. Wiley (1995)

[35] Khoo, I.C., Slussarenko, S., Guenther, B.D., Shih, M.Y., Chen, P., Wood, W.V.: Optically induced space-charge fields, dc voltage, and extraordinarily large nonlinearity in dye-doped nematic liquid crystals. Optics Letters **23**(4) (1998) 253–255

[36] Chandrasekhar, S.: Columnar, discotic nematic and lamellar liquid crystals: Their structure and physical properties. In: Handbook of Liquid Crystals. Volume 2B. Wiley-VCH (1998) 749–780

[37] Crossland, W.A., Wilkinson, T.D.: Nondisplay applications of liquid crystals. In: Handbook of Liquid Crystals. Volume 1. Wiley-VCH (1998) 763–822

[38] Wright, P.V., Chambers, B., Barnes, A., Lees, K., Despotakis, A.: Progress in smart microwave materials and structures. Smart Materials and Structures **9** (2000) 272–279

[39] Mortimer, R.J.: Electrochromic materials. Chemical Society Reviews **26** (1997) 147–156

[40] Bar-Cohen, Y.: Electroactive Polymer (EAP) Actuators as Artificial Muscles - Reality, Potential and Challenges. SPIE Press (2001)

[41] Pope, M., Swenberg, C.E.: Electronic Processes of Organic Crystals and Polymers. Oxford University Press (1999)

[42] Hao, T.: Electrorheological Fluids: The Non-aqueous Suspensions. Elsevier Science (2005)

[43] Khusid, B., Activos, A.: Effects of interparticle electric interactions on dielectrophoresis in colloidal suspensions. Physical Review E **54**(5) (1996) 5428–5435

[44] Khusid, B., Activos, A.: K. d. hermanson and s. o. lumsdon and j. p. williams and e. w. kaler and o. d. velev. Science **294** (2001) 1082–1086

[45] Petty, M.C.: Langmuir-Blodgett Films: An Introduction. Cambridge University Press (1996)

[46] Mills, J.W.: Polymer processors. Technical Report TR580, Department of Computer Science, University of Indiana (1995)

[47] Mills, J.W., Beavers, M.G., Daffinger, C.A.: Lukasiewicz logic arrays. Technical Report TR296, Department of Computer Science, University of Indiana (1989)

[48] Mills, J.W.: Programmable vlsi extended analog computer for cyclotron beam control. Technical Report TR441, Department of Computer Science, University of Indiana (1995)

[49] Mills, J.W.: The continuous retina: Image processing with a single sensor artificial neural field network. Technical Report TR443, Department of Computer Science, University of Indiana (1995)

[50] 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). Volume 2. (2004) 1800–1807

[51] Crooks, J.: Evolvable analogue hardware. Meng project report, The University Of York (2002)

[52] Lloyd, S.: Ultimate physical limits to computation. Nature **406** (2000) 1047 – 1054

[53] Koza, J.R.: Human-competitive machine intelligence by means of genetic algorithms. In Booker, L., Forrest, S., Mitchell, M., Riolo, R., eds.: Festschrift in honor of John H. Holland, Ann Arbor, MI: Center for the Study of Complex Systems (1999) 15–22

[54] Penrose, R.: The Emperor's New Mind, Concerning Computers, Minds, and the Laws of Physics. Oxford University, Oxford, UK (1989)

[55] Raichman, N., Ben-Jacob, E., Segev, R.: Evolvable hardware: Genetic search in a physical realm. In: Physica A. Volume 326. (2003) 265–285

[56] Siegelmann, H.T.: Neural Networks and Analog Computation, Beyond the Turing Limits. Birkhauser, Boston, MA (1999)

[57] Sienko, T., Adamatzky, A., Rambidi, N., Conrad, M.: Molecular Computing. MIT Press, Cambridge, MA (2003)

[58] Sinha, S., Munakata, T., Ditto, W.L.: Flexible parallel implementation of logic gates using chaotic elements. In: Physical Rev. Volume E, 65, 036216. (2002)