Evolution-In-Materio: A Frequency Classifier Using Materials

Maktuba Mohid[1], Julian F. Miller[1], Simon L. Harding[1], Gunnar Tufte[2], Odd Rune Lykkebø[2], Mark Kieran Massey[3], and Michael C. Petty[3]

[1]Department of Electronics, University of York, York, UK. Emails: [mm1159, julian.miller]@york.ac.uk, slh@evolutioninmaterio.com

[2]Department of Computer and Information Science, Norwegian University of Science and Technology, 7491 Trondheim, Norway. Emails: [gunnart, lykkebo]@idi.ntnu.no

[3]School of Engineering and Computing Sciences and Centre for Molecular and Nanoscale Electronics, Durham University, UK. Emails: [m.k.massey, m.c.petty]@durham.ac.uk

***Abstract*—Evolution-in-materio (EIM) is a method that uses artificial evolution to exploit properties of materials to solve computational problems without requiring a detailed understanding of such properties. In this paper, we describe experiments using a purpose-built EIM platform called Mecobo to classify whether an applied square wave signal is above or below a user-defined threshold. This is the *first demonstration* that electrical configurations of materials (carbon nanotubes and a polymer) can be evolved to act as frequency classifiers.**

I. Introduction

One way to view natural evolution is that it is an algorithm for exploiting the physical properties of materials, particularly proteins. Evolution-in-materio (EIM) aims to mimic this by manipulating physical systems using computer controlled evolution (CCE) [7], [11]. For a recent view of EIM see [12]. So far EIM has been mainly used to exploit the properties of physical systems for solving computational problems.

Evolution-in-materio was first described by Miller and Downing [11]. The concept was inspired by the work of Adrian Thompson who investigated whether it was possible for unconstrained evolution to evolve working electronic circuits using a silicon chip called a Field Programmable Gate Array (FPGA). He evolved a digital circuit that could discriminate between 1kHz or 10kHz signal [15]. When the evolved circuit was analysed, Thompson made a surprising discovery: artificial evolution had exploited some of the physical properties of the chip. After Thompson's work, a number of researchers investigated the ability of artificial evolution to exploit various physical properties of silicon-based integrated circuits (see the review [12]). In 2004, Harding and Miller demonstrated that the physical properties of non-silicon materials could also be exploited using EIM [5]. The configurable computational material chosen was liquid crystal. Subsequently they found that computer-controlled evolution could utilize the physical properties of liquid crystal to help solve a number of computational problems:

- Two input logic gates: OR, AND, NOR, NAND, etc. [7].
- Tone Discriminator: A device was evolved which could differentiate different frequencies [5].
- Robot Controller: A controller for a simulated robot with wall avoidance behavior [6].

In this paper, we describe the use of a purpose built platform called Mecobo that facilitates the evolution of electrical configurations of materials for solving a variety of computational problems. Mecobo is described in detail in [9]. The platform has been specifically developed within an EU funded research project called NASCENCE [3].

In this paper, we show that using the Mecobo platform, it is possible to evolve electrical configurations of materials to classify frequencies using a variety of user-defined threshold values. In these investigations the computational material is a mixture of single-walled carbon nanotubes and a polymer.

Although EIM has been previously shown to be able to discriminate between pairs of applied square-waves of different frequencies, we report for the first time in this paper, that EIM can be used to classify frequencies. Our aim is not to claim EIM is necessarily particularly suited to classifying frequencies, but rather we are simply trying to apply EIM to standard problem so that we have a yardstick to assess various aspects of EIM using the Mecobo platform. We are aware that using materials in the genotype-phenotype map within an evolutionary process has, at present, some drawbacks. The main one is that it is slow (see later) this means that we can only feasibly evaluate relatively few potential solutions. However, it is a new approach to the solution of computational problems and as the technology is developed it could offer advantages over conventional computational methods [12].

The organization of the paper is as follows. In Sect. II we give a conceptual overview of EIM. We describe the Mecobo EIM hardware platform in Sect. III. The preparation and composition of the physical computational material is described in Sect. IV. Description of frequency classification problem and the way we have used the Mecobo platform for classifying frequency is described in Sect. V. We describe our

experiments and analysis of results in Sect. VI. Finally we conclude and offer suggestions for further investigation in Sect. VII.

II. Conceptual Overview Of Evolution-In-Materio

EIM is a hybrid system involving both a physical material and a digital 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 [12]. Figure 1 shows conceptual overview of EIM.

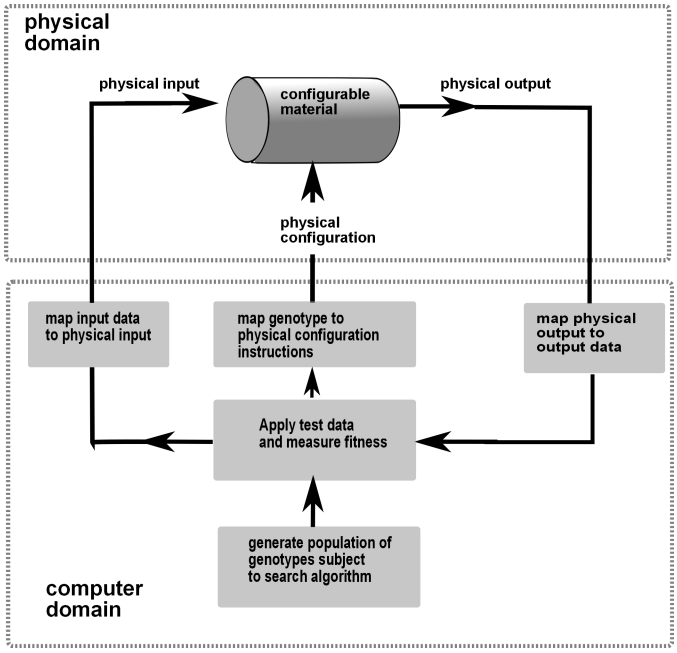


Fig. 1: Concept of evolution-in-materio [12].

In EIM a highly indirect genotype-phenotype mapping is employed. One of its interesting features is that an evolutionary algorithm may be able to exploit hitherto unknown physical variables in a material which may increase evolvability. Software-only genotype-phenotype mappings are highly constrained. Natural evolution operates in a physical world and exploits the physical properties of materials (mainly proteins). In an analysis of how evolutionary computation could be enriched Banzhaf et al. identified that physicality and embodiment are an important factor in computational evolution [2]. Despite this, there have been very few attempts to date, to include materials in the evolutionary process. THE NASCENCE project aims to remedy this situation.

Defining what materials may be suitable for EIM is not easy and Miller and Downing suggested some guidelines for choosing materials. The material needs to be reconfigurable, i.e., it should be able to be evolved over many configurations to get desired response. It is probably advantageous for the physical material to be able to be "reset" in some way before applying new input signals on it, otherwise it might preserve some memory and might give fitness scores that are dependent on the past behaviour. Preferably the material should be physically configured using small voltage and be manipulable at a molecular level [11], [12].

III. Mecobo: an Evolution-In-Materio Hardware Platform

Mecobo has been designed to interface a large variety of materials. The hardware allows for the possibility to map input, output and configuration terminals, to alter signal properties in many ways and also offers highly flexible monitoring of outputs. The platform's software components, (i.e.the EA and the software stack) are as important as the hardware. A flexible software platform is built into Mecobo which supports multiple programming languages and also the possibility to connect to hardware over the internet [9].

It is important to appreciate that in EIM the computational substrate is piece of material. It is expected that in this case appropriate physical variables to be manipulated by evolution will most probably be poorly understood (see Fig 1). Consequently, the platform must be free to select of signal types, i.e. inputs, outputs and configuration data, and their assignment to the I/O ports. In addition, the signal properties, e.g. voltage/current levels, AC, DC, pulse or frequency, should be allowed to be chosen during evolution. The Mecobo hardware interface is designed to handle all these features. Many computational problems require input data so the Mecobo interface has been designed to allow user-defined external input data signals.

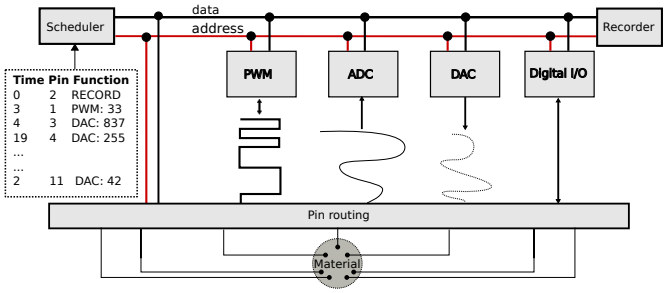


Fig. 2: Overview of the complete system.

Figure 2 shows an overview of the hardware interface. In the figure an example set up is shown in the dotted box. The example genome defines pin 2 to be the output terminal, pin 1 to be the data input and pin 3 - 12 to be configuration signals. The architecture is controlled by a scheduler controlling the

following modules: Digital I/O can output digital signals and sample responses. Analogue output signals can be produced by the DAC module. The DAC can be configured to output static voltages or any arbitrary time dependent waveform. Sampling of analogue waveforms from the material is performed by the ADC. Pulse Width Modulated (PWM) signals are produced by the PWM module.

The system's scheduler can set up the system to apply and sample signals statically or produce time scheduled configurations of stimuli/response. The recorder stores samples, digital discrete values, time dependent bit strings, sampled analogue discrete values or time dependent analogue waveforms. Note that the recorder can include any combination of these signals.

In the interface all signals pass a crossbar, i.e. pin routing. Pin routing is placed between the signal generator modules and the sampling buffer (PWM, ADC, DAC, Digital I/O and Recorder) making it possible to configure any terminal of a material to be input, output or receive configuration signals.

The material signal interface presented in Figure 2 is very flexible. It not only allows the possibility to evolve the I/O terminal placement but also allows a large variety of configuration signals to be used, from static signals to time dependent digital functions. At present, the response from materials can be sampled as purely static digital signals, digital pulse trains. The next version of Mecobo will allow the input and output of analogue signals. Further the scheduler can schedule time slots for different stimuli when time dependent functions are targeted or to compensate for configuration delay, i.e. when materials need time to settle before a reliable computation can be observed.

A. Hardware implementation

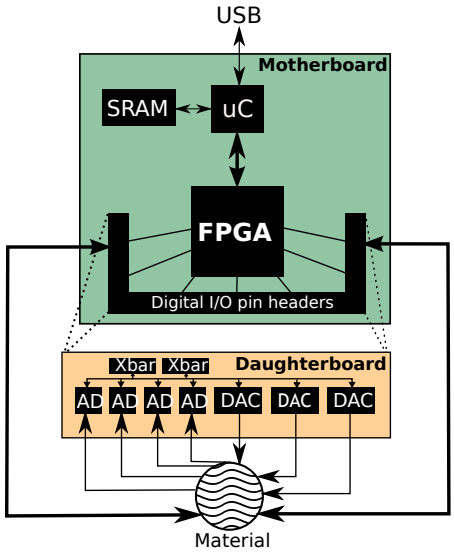
The hardware implementation of the interface is shown as a block diagram in figure 3(a). Mecobo is implemented on a PCB with an FPGA as the main component. The system shown in Figure 2 is part of the FPGA design together with communication modules interfacing a micro controller and shared memory. As shown in Figure 3(a) the digital and analogue designs are split into two. All analogue components are placed on a daughter board; such as crossbar switches and analogue-digital converters. This allows the redesign of the analogue part of the system without changing the digital part of the motherboard. The system shown in Figure 3(a) is an example of the current system. The micro controller stands as a communication interface between the FPGA and the external USB port.

Figure 3(b) shows the motherboard with the Xilinx LX45 FPGA, Silicon Labs ARM based EFM32GG990 micro controller connected to a 12 terminal material sample.

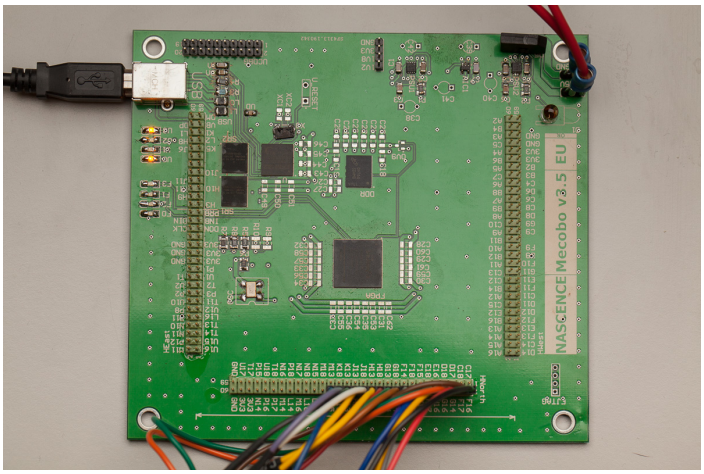
At present the Mecobo hardware allows only two types of inputs to the material: constant voltage (0V or 3.5V) or a square wave signal. However, different characteristics or input parameters associated with these inputs can be chosen. These input parameters are described in Table I.

The start time and end time of each input signal determines how long an input is applied.

In addition, when electrodes are read, a user-defined output sampling frequency determines the buffer size of output



(a) Mecobo block diagram.



(b) Picture of Mecobo.

Fig. 3: Hardware interface implementation overview.

samples. The size of the buffer is determined by the sampling frequency and the time the electrode is being read. This is shown in Eqn. 1.

If the output frequency is F_{out}, start time $Time_{start}$ and end time is $Time_{end}$, then the buffer size is Buf_{size} is given by:

$$Buf_{size} = F_{out}(Time_{end} - Time_{start})/1000 \qquad (1)$$

Here, $Time_{start}$ and $Time_{end}$ are measured in milliseconds. However, in practice due to pin latency, the real buffer size is generally smaller.

TABLE I: Adjustable Mecobo input parameters.

Parameter Name	Description	Note
Amplitude	0 or 1 corresponding to 0V or 3.5V	wave signal amplitude must be 1
Frequency	Frequency of square wave signal	Irrelevant if fixed voltage input
Cycle Time	Percentage of period for which square wave signal is 1	Irrelevant if fixed voltage input
Phase	Phase of square wave signal	Irrelevant if fixed voltage input
Start time	Start time of applying voltage to electrodes	Measured in milliseconds.
End time	End time of applying voltage to electrodes	Measured in milliseconds.

IV. Description Of Physical Computational Material

The experimental material we used consists of single-walled carbon nanotubes (SWCNTs) mixed with poly(methyl methacrylate) (PMMA), dissolved in Anisole (methoxybenzene) [1]. This mixture is mixed using an ultrasonic homogenizer and then a small volume (20 μL) is spread on a gold microelectrode array. The sample is baked to evaporate the solvent, leaving behind a film of SWCNT and PMMA. The concentration of SWCNT is 0.71 % (expressed as a weight % fraction of the PMMA).

Carbon nanotubes, as grown, contain a mixture of semiconducting and metallic species. The role of the PMMA in these composites is to introduce insulating regions within the SWCNT network, creating non-linear current versus voltage characteristics. The idea is that this might show some interesting computational behaviour. Another benefit of the polymer is to help with dispersion of the nanotubes in solution. The process of preparing experimental material is given below:

- A M3-sized nylon washer was glued to the electrode array to contain the material whilst drying;
- 20 μL of material was dispensed into the washer;
- This was dried at 100^o C for 1 hr leaving a "thick film".

The electrode array consisted of 12 electrode tracks, connected to the SWCNT/PMMA material. The electrode array is connected directly with the Mecobo board via wires. The electrode sample is shown in Fig. 4.

[1]Mark K. Massey and Michael C. Petty prepared the materials used as substrates and the electrode masks for our experiments.

Fig. 4: Electrode array with sample.

V. Classifying Frequency Using Evolution-In-Materio

The idea of a frequency classifier is to classify whether an applied frequency is above or below a user-defined threshold. Thus all frequencies less than or equal to the threshold belong to one class and frequencies higher belong to the other class.

A. Methodology

The experiments were performed with an electrode array having 12 electrodes. One electrode was used to input the signal to be classified, 2 electrodes were used as outputs and 9 electrodes were used as configuration voltages. Each chromosome defined which electrodes were either outputs, inputs (receive square waves) or received the configuration data (square waves or constant voltage). The cycle time of input electrode was set to 50% and its amplitude was set to one (i.e. 3.5 V).

Using the Mecobo platform we can control the time (in milliseconds) that a signal is applied to the material (see Sect. III). Here, we accumulated output values in a buffer for 128 milliseconds. The number of samples stored in the output buffer can be controlled by the start time, end time and the sampling frequency of output electrode. In experiment, we used a 25KHz buffer sampling frequency.

The frequency classification was interpreted as two class problem. Each output was associated with a particular class.

In the evaluation of each chromosome we used ten input frequencies: 1KHz-10KHz in 1KHz intervals. At the end of each evolutionary run the final evolved configuration of electrodes was tested using 10 different input frequencies: 0.5KHz - 9.5KHz in 1 KHz intervals. In each evolutionary run and test run, a fixed input threshold was used. We did experiments with seven different thresholds and they were: 1.375KHz, 2.750KHz, 4.125KHz, 5.500KHz, 6.875KHz, 8.250KHz, 9.625KHz.

B. Genotype Representation

Each chromosome used $n_e = 12$ electrodes at a time. Associated with each electrode there were six genes which define which electrode was used as an input or output or configuration voltage, or characteristics of the input applied to the electrode: signal type, amplitude, frequency, phase, cycle (see Sect. III). This means that each chromosome required a total of 72 genes. Mutational offspring were created from a parent genotype by mutating a single gene (i.e one gene of 72). The values that genes could take are shown in Table II where i takes values 0, 1, ... 11.

TABLE II: Description of genotype.

Gene Symbol	Signal applied to, or read from i^{th} electrode	Allowed values
p_i	Which electrode is used	0, 1, 2 ... 11
s_i	Type	0 (constant) or 1(square-wave)
a_i	Amplitude	0 , 1
f_i	Frequency	500 ,501 ... 10K
ph_i	Phase	1, 2 ... 10
c_i	Cycle	0, 1, ... 100

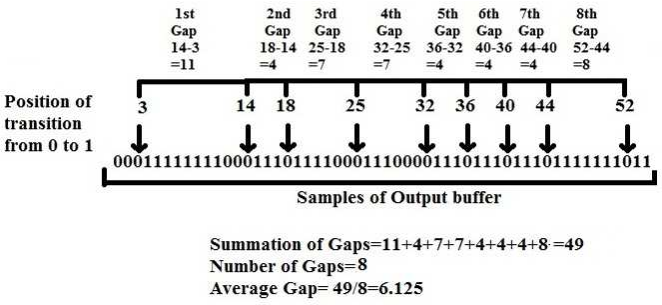


Fig. 5: Example of average transition gap calculation for an output electrode.

The genotype for a chromosome of an individual consists of the 72 genes shown below:

$$p_0 s_0 a_0 f_0 ph_0 c_0 \ldots p_{11} s_{11} a_{11} f_{11} ph_{11} c_{11}$$

C. Output Mapping

We determined the output, by examining the output buffers which contain samples taken from the output electrodes. Since the current Mecobo platform can only recognize binary values, the output buffers contain bitstrings. We used the *transitions* from 0 to 1 in the output buffers to define the output. For each output buffer the positions of transitions were recorded and the gaps between consecutive transitions were measured and an average calculated. A transition based fitness was used as it is frequency related. An example of average gap calculation for an output electrode is shown in Fig. 5

The class associated with an output electrode was determined by the output buffer with lower average transition gap. If the first electrode had lower average transition gap, it was designated to be class one, otherwise it was designated to be class two. So, the first output electrode was expected to have lower average transition gap only if the input frequency was less than or equal to threshold.

Thus, if the frequency classifier works as desired, it would have class one, when the first electrode has a lower average transition gap whenever the input frequency is lower or equal to the threshold. It would have class two, when the first electrode has a higher average transition gap at the time when input frequency is higher than threshold.

D. Fitness Score

The fitness calculation required counts to be made of the number of true positives TP, true negatives TN, false positives, FP and false negatives, FN. There are four possible cases.

- If input frequency is lower or equal to threshold and first electrode has lower average transition gap, it is correct behavior, in this case TP = TP + 1;
  TN = TN + 1;

- If input frequency is higher than threshold and second electrode has lower average transition gap, it is correct behavior, in this case TP = TP + 1;
  TN = TN + 1;

- If input frequency is lower or equal to threshold and second electrode has lower average transition gap, it is incorrect behavior, in this case FP = FP + 1;
  FN = FN + 1;

- If input frequency is higher than threshold and first electrode has lower average transition gap, it is incorrect behavior, in this case FP = FP + 1;
  FN = FN + 1;

In classification problems the set of value TP, TN, FP, FN accumulated over all instance data (in our cases different applied frequencies) define the so-called confusion matrix. To obtain the fitness value we employed the Matthew's correlation coefficient (MCC). The MCC is recognized as one of the best single number measures of the quality of a classification algorithm based on the confusion matrix [1]. It can be applied to balanced or unbalanced datasets. The MCC is calculated using Eqn. 2 and was the fitness function adopted in our experiments.

$$MCC = \frac{TP.TN - FP.FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}} \quad (2)$$

If all results are correct, the fitness is 1, since in this case $FP = 0$ and $FN = 0$. In the case that all results are incorrect, then $TP = 0$ and $TN = 0$, so fitness is -1.

It can be shown that TN has the same value as TP since when frequencies are correctly classified, both TP and TN are incremented by same amount (see the beginning of this section). Similarly FN has same value as FP as when frequencies are incorrectly classified, both FP and FN are incremented by same amount. If we replace TN by TP and FN by FP, in Eqn. 2, we get,

$$MCC = \frac{TP - FP}{TP + FP} \quad (3)$$

As either the device has the correct response to a square wave of a particular frequency or not, and in each case either

TP has been incremented by one or TN has been incremented by one. The sum must be equal to the number of frequencies applied. This is ten in both training and test situations. Thus we have Eqn. 4.

$$TP + FP = 10 \tag{4}$$

Solving equations 3 and 4, we get:

$$TP = 5(MCC + 1) \tag{5}$$

And, the number of correctly classified frequencies, N_c is same as the value of TP, thus $N_c = 5(MCC + 1)$, so for instance, if $MCC = 0.8$, nine out of ten input frequencies are correctly classified.

VI. Experiments

To evaluate each chromosome we applied the ten input frequencies and measured the response. We also carried out collections of evolutionary runs using seven different thresholds.

For each of these experiments a $1 + \lambda - ES$, evolutionary algorithm with $\lambda = 4$ was used [10]. This simple evolutionary algorithm has a population size of $1 + \lambda$ and selects the genotype with the best fitness to be the parent of the new population. If there is no offspring with a better fitness than the parent but there is an offspring with a fitness equal to the parent, then that offspring is chosen to be the new parent. The remaining members of the population are formed by mutating the parent. This algorithm was used partly because of its simplicity and partly because in other studies comparisons have been made between an evolutionary software approach and evolution-in-materio[4], [14], [13].

The evolutionary algorithm was run for a maximum of 5000 generations. However, evolutionary runs were terminated if fitness score reached at value 1.0 (when all the results for all input frequencies were correct).

Twenty independent runs were carried out for all of these experiments. The total time required for all 20 runs of all experiments was almost 4 days.

A. Analysis of Results

The experiments show that in case of all of these 7 experiments, the average results of evolutionary run have accuracy 100% and standard deviation 0.0. But the average testing accuracy of all 20 runs is not 100%. But the best testing accuracy of all 20 runs is 100% in case of all 7 experiments. For details see table III.

Fig. 6 shows the change in fitness in different generations for five evolutionary runs using a threshold 2.750KHz.

Inspections were carried out on all the final gene values of configuration data of one frequency classifier problem to see if there was a common pattern, however none was found. The data of one run of one frequency classifier problem (having threshold 5500 Hz and both training and testing accuracy are 100%) has been shown in table IV.

One of the frequency classifier experiments (using a threshold 4125Hz) was performed for a single evolutionary run of 100 generations using an empty electrode array. This experiment was intended to investigate how important the material is for the evolution of a working classifier. It was found that no evolution was possible with an empty electrode. The fitness values of any configurations were found to be zero. This remained the same after 100 generations. Thus we can conclude that the computation performed to solve all the frequency classification problems require the physical material. However, this does not in itself decide whether part of the calculation happens in the Mecobo platform itself, or whether it is purely in the material alone. Further work is required to answer this question.

TABLE III: Experimental results with different input thresholds. The second column shows average MCC for test data over all 20 runs.

Input Threshold	Average Testing Result (Matthew's coefficient)
1375Hz	0.79
2750Hz	0.79
4125Hz	0.61
5500Hz	0.53
6875Hz	0.66
8250Hz	0.62
9625Hz	0.85

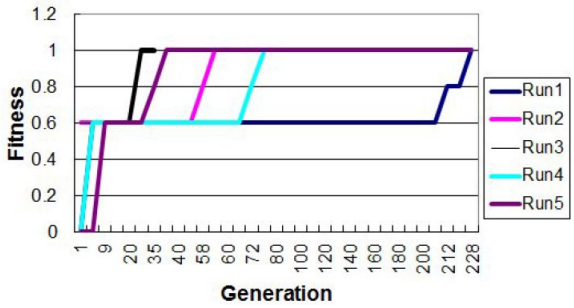


Fig. 6: Fitness vs. number of generations for five evolutionary runs using a threshold of 2.750KHz.

B. Stability of Results

To investigate whether the behaviour of the experimental material was stable and reliable we re-applied the final evolved configuration of electrodes from some runs of some experiments and measured the response of the material under different circumstances. These are described below:

- Signals were applied to electrodes, then Mecobo was stopped and started again, then the same signals were re-applied to same electrodes.
- Signals were applied to electrodes, then Mecobo was left idle for long time being turned off, then the same signals were re-applied to the same electrodes.
- Signals were applied to electrodes, then some random signals were applied and then subsequently the original signals were re-applied to same electrodes.

- Signals were applied to electrodes twice one after the other.

In almost all cases we obtained completely stable results. Occasionally there was a small difference in response which meant that a single frequency might be misclassified. We also repeated our experiment many times and examined if there were significant variations in the calculated average transition gaps of two electrodes. We found very little variation. We also observed that in different evolutionary runs under the same conditions, the average transition gaps varied little.

To further investigate stability we looked at other types of output measurement. Using a single output electrode we measured the percentages of ones in an output buffer. Once again we carried out evolutionary runs and varied the environments as before. We found very little variation in percentages of ones in the output buffers. We also examined the behaviour on re-applying evolved configurations after intervals of months and obtained very little change in behaviour from what happened on the first configuration.

In a summary we find that the evolved behaviors are very stable. Further experiments would be needed to quantify exactly what small variations occur.

VII. Conclusions And Future Outlook

Evolution-in-materio is a form of hybrid digital-analogue computing where digital computers can help configure materials to carry out analogue computation. Using this technique, it may be possible to develop entirely new computational devices. A purpose-built evolutionary platform called Mecobo, has been used to evolve configurations of a physical system to classify frequencies. The material used is a mixture of single-walled carbon nanotubes and a polymer. The aim of the paper is not to show that the experimental results of classifying frequencies using EIM is competitive with any other frequency classification methods, but rather to show that materials can be manipulated to solve problem by computer-controlled evolution without requiring detailed knowledge or models of the materials' underlying behaviour. This is the first time it has been shown that such an approach can be used to classify frequencies. It should be noted that in all of the cases, we found that we could find frequency classifiers which could give results with accuracy 100%. In other work using Mecobo it has been shown that digital logic functions can be implemented [9]. We have also obtaining encouraging results on function optimization [13], bin packing, traveling salesman [4] and machine learning classification [14] problems. In principle, some evolved systems could act as standalone devices. This could have utility in robot control and other applications.

There remain many questions for the future. How does evolutionary computation in materio scale on larger problem instances. What other classes of computational problems can be solved using this technique? What are the most suitable materials and signal types for evolution-in-materio? The Mecobo platform is currently under development and the next version will be able to allow the utilization of analogue voltages. This may make some types of computational problems more readily solved using evolution-in-materio. The material we have used does not change physically after the application of physical signals, but rather it is configured electronically (rather like silicon in conventional electronics), in future work we will investigate other materials (e.g. CNTs in liquid crystal). In these materials it is possible to change the orientation and position of carbon nanotubes through the action of applied fields. If such material configurations could be evolved and made stable then it may be possible to construct standalone devices that operate at very low power.

How can we quantify the computational capacity of such materials? Interestingly, Seth Lloyd calculated what the laws of physics could in principle allow [8], and calculated that the ultimate laptop would weigh one kilogram and could theoretically carry out 10^{50} logical operations per second. However, it would operate at a temperature of 10^9K!

VIII. Acknowledgements

The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement number 317662.

References

[1] Baldi, P., Brunak, S., Chauvin, Y., Andersen, C.A.F., Nielsen, H.: Assessing the accuracy of prediction algorithms for classification: An overview. Bioinformatics **16**, 412 – 424 (2000)

[2] Banzhaf, W., Beslon, G., Christensen, S., Foster, J., Képès, F., Lefort, V., Miller, J., Radman, M., J., R.: Guidelines: From artificial evolution to computational evolution: a research agenda. Nature Reviews Genetics **7**, 729–735 (2006)

[3] 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)

[4] Clegg, K.D., Miller, J.F., Massey, K., Petty, M.: Travelling salesman problem solved 'in materio' by evolved carbon nanotube device. In: T. Bartz-Beielstein, J. Branke, B. Filipič, J. Smith (eds.) Parallel Problem Solving from Nature PPSN XIII, *Lecture Notes in Computer Science*, vol. 8672, pp. 692–701. Springer International Publishing (2014)

[5] 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)

[6] 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)

[7] 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)

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

[9] Lykkebø, O.R., Harding, S., Tufte, G., Miller, J.F.: Mecobo: A hardware and software platform for in materio evolution. In: O.H. Ibarra, L. Kari, S. Kopecki (eds.) Unconventional Computation and Natural Computation, LNCS, pp. 267–279. Springer International Publishing (2014)

[10] Miller, J.F. (ed.): Cartesian Genetic Programming. Springer (2011)

[11] Miller, J.F., Downing, K.: Evolution in materio: Looking beyond the silicon box. In: A. Stoica, J. Lohn, R. Katz, D. Keymeulen, R.S. Zebulum (eds.) The 2002 NASA/DoD Conference on Evolvable Hardware, vol. 7, pp. 167 – 176. IEEE Computer Society (2002)

TABLE IV: Final gene values of configuration voltages in one run of classifier problem using threshold 5500 Hz. This run achieved training and testing accuracies of 100%. 'n/a' is used when frequency, phase and cycle values are not applicable, i. e., for constant voltage.

Configuration voltage	Electrode no.	Signal type	Amplitude	Frequency (Hz)	Cycle	Phase
1	3	Wave	0	4755	85	7
2	9	Constant	1	n/a	n/a	n/a
3	8	Wave	0	4528	16	9
4	1	Wave	1	7026	34	7
5	11	Constant	0	n/a	n/a	n/a
6	10	Constant	1	n/a	n/a	n/a
7	0	Wave	1	3112	77	2
8	2	Wave	1	2422	31	8
9	6	Constant	0	n/a	n/a	n/a

[12] Miller, J.F., Harding, S.L., Tufte, G.: Evolution-in-materio: evolving computation in materials. Evolutionary Intelligence **7**, 49–67 (2014)

[13] Mohid, M., Miller, J.F., Harding, S.L., Tufte, G., Lykkebø, O.R., Massey, M.K., Petty, M.C.: Evolution-in-materio: Solving function optimization problems using materials. In: Computational Intelligence (UKCI), 2014 14th UK Workshop on, pp. xx–xx (2014). In press

[14] Mohid, M., Miller, J.F., Harding, S.L., Tufte, G., Lykkebø, O.R., Massey, M.K., Petty, M.C.: Evolution-in-materio: Solving machine learning classification problems using materials. In: T. Bartz-Beielstein, J. Branke, B. Filipič, J. Smith (eds.) Parallel Problem Solving from Nature PPSN XIII, *Lecture Notes in Computer Science*, vol. 8672, pp. 721–730. Springer International Publishing (2014)

[15] Thompson, A.: Hardware Evolution - Automatic Design of Electronic Circuits in Reconfigurable Hardware by Artificial Evolution. Springer (1998)