ENOC - 2005

Fifth EUROMECH
Nonlinear Dynamics Conference

Fifth EUROMECH
Nonlinear Dynamics Conference
August 7 -12, 2005

Chair Prof. Dick H. van Campen

Eindhoven University of Technology
Auditorium Building, The Netherlands

ENOC-2005, Eindhoven, Netherlands, 7-12 August 2005

EVOLUTIONARY DESIGN OF COMBINATIONAL CIRCUITS USING FRACTIONAL-ORDER FITNESS FUNCTIONS

Cecília Reis [1], **J. A. Tenreiro Machado**[1]
[1]Institute of Engineering of Porto
Polytechnic Institute of Porto
Portugal
{cecilia, jtm}@dee.isep.ipp.pt

J. Boaventura Cunha[2]
[2] Engineering Department
University of Trás-os-Montes and Alto Douro
Portugal
jboavent@utad.pt

Abstract
This paper analyses the performance of a genetic algorithm using the new concept of fractional-order dynamic fitness function, for the synthesis of combinational logic circuits. The experiments reveal superior results in terms of speed and convergence to achieve a solution.

Key words
Circuit Design, Fractional-Order Systems, Genetic Algorithms, Logic Circuits.

1 Introduction

In the last decade genetic algorithms (GAs) have been applied in the design of electronic circuits, leading to a novel area of research called Evolutionary Electronics (EE) or Evolvable Hardware (EH) [Zebulum et al., 2001]. EE considers the concept for automatic design of electronic systems. Instead of using human conceived models, abstractions and techniques, EE employs search algorithms to develop good designs [Thompson and Layzell, 1999].

One decade ago Sushil and Rawlins [Louis and Rawlins, 1991] applied GAs to the combinational circuit design problem. They combined knowledge-based systems with the GA and defined a genetic operator called masked crossover. This scheme leads to other kinds of offspring that can not be achieved by classical crossover operators.

John Koza [Koza, 1992] adopted genetic programming to design combinational circuits.

In the sequence of this work, Coello, Christiansen and Aguirre [Coello et al., 1996] presented a computer program that automatically generates high-quality circuit designs. They use five possible types of gates (AND, NOT, OR, XOR and WIRE) with the objective of finding a functional design that minimizes the use of gates other than WIRE.

Miller, Thompson and Fogarty [Miller et al., 1997] applied evolutionary algorithms for the design of arithmetic circuits. The technique was based on evolving the functionality and connectivity of a rectangular array of logic cells, with a model of the resources available on the Xilinx 6216 FPGA device.

Kalganova, Miller and Lipnitskaya [Kalganova et al., 1998] proposed a new technique for designing multiple-valued circuits.

In order to solve complex systems, Torresen [Torresen, 1998] proposed the method of increased complexity evolution. The idea is to evolve a system gradually as a kind of divide-and-conquer method. Evolution is first undertaken individually on simple cells. The evolved functions are the basic blocks adopted in further evolution of more complex systems.

A major bottleneck in the evolutionary design of electronic circuits is the problem of scale. This refers to the very fast growth of the number of gates, used in the target circuit, as the number of inputs of the evolved logic function increases. This results in a huge search space that is difficult to explore even with evolutionary techniques. Another related obstacle is the time required to calculate the fitness value of a circuit [Vassilev and Miller, 2000]. A possible method to solve this problem is to use building blocks either than simple gates. Nevertheless, this technique leads to another difficulty, which is how to define building blocks that are suitable for evolution.

Timothy Gordon [Gordon and Bentley, 2002] suggests an approach that allows evolution to search for good inductive bases for solving large-scale complex problems. This scheme generates, inherently, modular and iterative structures, that exist in many real-world circuit designs but, at the same time, allows evolution to search innovative areas of space.

The idea of using memory to achieve better fitness function performances was first introduced by Sano and Kita [Sano and Kita, 2000]. Their goal was the optimization of systems with randomly fluctuating fitness function and they developed a Genetic Algorithm with Memory-based Fitness Evaluation (MFEGA). The key ideas of the MFEGA are based on storing the sampled fitness values into memory as a search history, introducing a simple stochastic model of fitness values to be able to estimate fitness

values of points of interest using the history for selection operation of the GA.

Following this line of research, and looking for better performance GAs, this paper proposes a GA for the design of combinational logic circuits using fractional-order dynamic fitness functions.

The area of Fractional Calculus (FC) deals with the operators of integration and differentiation to an arbitrary (including noninteger) order and is as old as the theory of classical differential calculus [Oldham and Spanier, 1974, Miller and Ross, 1993]. The theory of FC is a well-adapted tool to the modelling of many physical phenomena, allowing the description to take into account some peculiarities that classical integer-order models simply neglect. Nevertheless, the application of FC has been scarce until recently, but the advances on the theory of chaos motivated a renewed interest in this field. In the last two decades we can mention research on viscoelasticity/damping, chaos/fractals, biology, signal processing, system identification, diffusion and wave propagation, electromagnetism and automatic control [Oustaloup, 1995, Méhauté, 1991, Machado, 1997, Westerlund, 2002].

Bearing these ideas in mind the article is organized as follows. Section 2 describes the adopted GA as well as the fractional-order dynamic fitness functions. Section 3 presents the simulation results and finally, section 4 outlines the main conclusions and addresses perspectives towards future developments.

2 The Adopted Genetic Algorithm

In this section we present the developed GA, in terms of the circuit encoding as a chromosome, the genetic operators and the static and dynamic fitness functions.

2.1 Problem Definition

To design combinational logic circuits it is adopted a GA strategy. The circuits are specified by a truth table and the goal is to implement a functional circuit with the least possible complexity. Two sets of logic gates have been defined, as shown in Table I, being Gset a the simplest one (*i.e.*, a RISC-like set) and Gset b a more complex gate set (*i.e.*, a CISC-like set).

For each gate set the GA searches the solution space, based on a simulated evolution aiming the survival of the fittest strategy. In general, the best individuals of any population tend to reproduce and survive, thus improving successive generations. However, inferior individuals can, by chance, survive and also reproduce. In our case, the individuals are digital circuits, which can evolve until the solution is reached (in terms of functionality and complexity).

Table I Gate sets

Gate Set	Logic gates
Gset a	{AND,XOR,WIRE}
Gset b	{AND,OR,XOR,NOT,WIRE}

2.2 Circuit enconding

In the GA scheme the circuits are encoded as a rectangular matrix A (row × column = $r \times c$) of logic cells as represented in figure 1.

Each cell is represented by three genes: <*input1*><*input2*><*gate type*>, where *input1* and *input2* are one of the circuit inputs, if they are in the first column, or, one of the previous outputs, if they are in other columns. The *gate type* is one of the elements adopted in the gate set. The chromosome is formed by as many triplets of this kind as the matrix size demands. For example, the chromosome that represents a 3 × 3 matrix is depicted in figure 2.

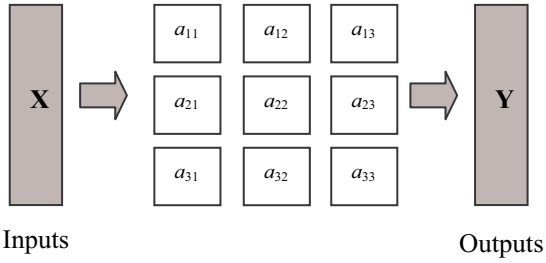


Figure 1: A 3 × 3 matrix A representing a circuit with input **X** and output **Y**.

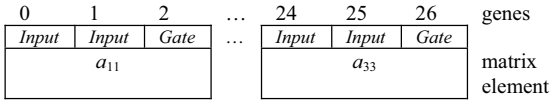


Figure 2: Chromosome for the 3 × 3 matrix of fig. 1.

2.3 The genetic operators

The initial population of circuits (strings) is generated at random. The search is then carried out among this population. The three different operators used are reproduction, crossover and mutation, as described in the sequel.

In what concern the reproduction operator, the successive generations of new strings are reproduced on the basis of their fitness function. In this case, it is used a tournament selection to select the strings from the old population, up to the new population.

For the crossover operator, the strings in the new population are grouped together into pairs at random. Single point crossover is then performed among pairs. The crossover point is only allowed between cells to maintain the chromosome integrity.

The mutation operator changes the characteristics of a given cell in the matrix. Therefore, it modifies

the gate type and the two inputs, meaning that a completely new cell can appear in the chromosome. Moreover, it is applied an elitist algorithm and, consequently, the best solutions are always kept for the next generation.

To run the GA we have to define the number of individuals to create the initial population *P*. This population is always the same size across the generations, until the solution is reached.

The crossover rate *CR* represents the percentage of the population *P* that reproduces in each generation. Likewise the mutation rate *MR* is the percentage of the population *P* that can mutates in each generation.

2.4 The Static and the Dynamic Fitness Functions

The goal of this study is to find new ways of evaluating the individuals of the population in order to achieve better performance GAs.

In this paper we propose two concepts for the fitness functions, namely the static fitness function F_s and the dynamic fitness function F_d.

The calculation of F_s in (1) is divided in two parts, f_1 and f_2, where f_1 measures the functionality and the error discontinuity and f_2 measures the simplicity. In a first phase, we compare the output **Y** produced by the GA-generated circuit with the required values $\mathbf{Y}_R$, according with the truth table, on a bit-per-bit basis. By other words, f_{11} is incremented by *one* for each correct bit of the output until f_{11} reaches the maximum value f_{10}, that occurs, when we have a functional circuit. After this, f_{11} is decremented by δ for each $\mathbf{Y_R} - \mathbf{Y}$ error discontinuity, where discontinuity means passing from $\mathbf{Y_R} - \mathbf{Y} = 0$ to $\mathbf{Y_R} - \mathbf{Y} = 1$ or vice-versa when comparing two consecutive levels of the truth table. Once the circuit is functional, in a second phase, the GA tries to generate circuits with the least number of gates. This means that the resulting circuit must have as much genes <*gate type*> ≡ <*wire*> as possible. Therefore, the index f_2, that measures the simplicity (the number of null operations), is increased by *one* (*zero*) for each *wire* (*gate*) of the generated circuit, yielding:

$$f_{10} = 2^{ni} \times no \tag{1a}$$

$$f_{11} = f_{11} + 1 \text{ if } \{\text{bit } i \text{ of } \mathbf{Y}\} = \{\text{bit } i \text{ of } \mathbf{Y_R}\} \tag{1b}$$

$$f_1 = f_{11} - \delta \text{ if error}_i \neq \text{error}_{i-1} \tag{1c}$$

$$f_2 = f_2 + 1 \text{ if } \textit{gate type} = \textit{wire} \tag{1d}$$

$$F_s = \begin{cases} f_1, & F_s < f_{10} \\ f_1 + f_2, & F_s \geq f_{10} \end{cases} \tag{1e}$$

Where $i = 1, \ldots, f_{10}$, *ni* and *no* represent the number of inputs and outputs of the circuit.

The concept of dynamic fitness function F_d results from an analogy with control systems where we have a variable to be controlled similarly with the GA case where we master the population through the fitness function. The simplest control system is the proportional algorithm; nevertheless, there can be other control algorithms, like the differential and the integral schemes. Therefore, applying the static fitness function corresponds to using a kind of proportional algorithm. If we want to implement a proportional-integral-derivative evolution the fitness function needs a scheme of the type:

$$F_d = F_s + K_I\, I^{\lambda} F_s + K_D\, D^{\mu} F_s \tag{2}$$

where $0.0 \leq \lambda \leq 1.0$ is the integral fractional-order, $0.0 \leq \mu \leq 1.0$ is the differential fractional-order and K_I, K_D are the integral and the differential 'gains' of the dynamical term, respectively.

The generalization of the concept of derivative $D^{\alpha}[f(x)]$ to noninteger values of α goes back to the beginning of the theory of differential calculus. In fact, Leibniz, in his correspondence with Bernoulli, L'Hôpital and Wallis, had several notes about its calculation for $\alpha = 1/2$ [Oldham and Spanier, 1974, Miller and Ross, 1993]. Nevertheless, the adoption of the *FC* in control algorithms has been recently studied using the frequency and discrete-time domains [Oustaloup, 1995, Méhauté, 1991, Machado, 1997].

The mathematical definition of a derivative of fractional order α has been the subject of several different approaches. For example, Eq. (3) and Eq. (4), represent the Laplace (for zero initial conditions) and the Grünwald-Letnikov definitions of the fractional derivative of order α of the signal $x(t)$:

$$D^{\alpha}[x(t)] = L^{-1}\{s^{\alpha} X(s)\} \tag{3}$$

$$D^{\alpha}[x(t)] = \lim_{h \to 0} \frac{1}{h^{\alpha}} \sum_{k=0}^{\infty} \frac{(-1)^k \Gamma(\alpha + 1)}{k!\,\Gamma(\alpha - k + 1)} x(t - kh) \tag{4}$$

where Γ is the gamma function and h is the time increment. This formulation [Machado, 1997] inspired a discrete-time calculation algorithm, based on the approximation of the time increment h through the sampling period T and a r-term truncated series yielding the equation:

$$D^{\alpha}[x(t)] \approx \frac{1}{T^{\alpha}} \sum_{k=0}^{r} \frac{(-1)^k \Gamma(\alpha + 1)}{k!\,\Gamma(\alpha - k + 1)} x(t - kT) \tag{5}$$

3 Experiments and Simulation Results

Reliable execution and analysis of a GA usually requires a large number of simulations to provide a reasonable assurance that stochastic effects have been properly considered. Therefore, in this study are developed $n = 1000$ simulations for each case.

The experiments consist on running the GA to

generate a typical combinational logic circuit, namely a 2-to-1 multiplexer (*M2-1*) and a 4-bit parity checker (*PC4*), using the fitness scheme described previously. The circuits are generated with the gate sets presented in Table 1 for $CR = 95\%$, $MR = 20\%$. $P = 100$ and the implementation of the differential/integral fractional order operator adopts Eq. (5) with a series truncation of $r = 50$ terms.

Having these ideas in mind, a superior GA performance means achieving solutions with a smaller number N of generations and a smaller standard deviation in order to reduce the stochastic nature of the algorithm.

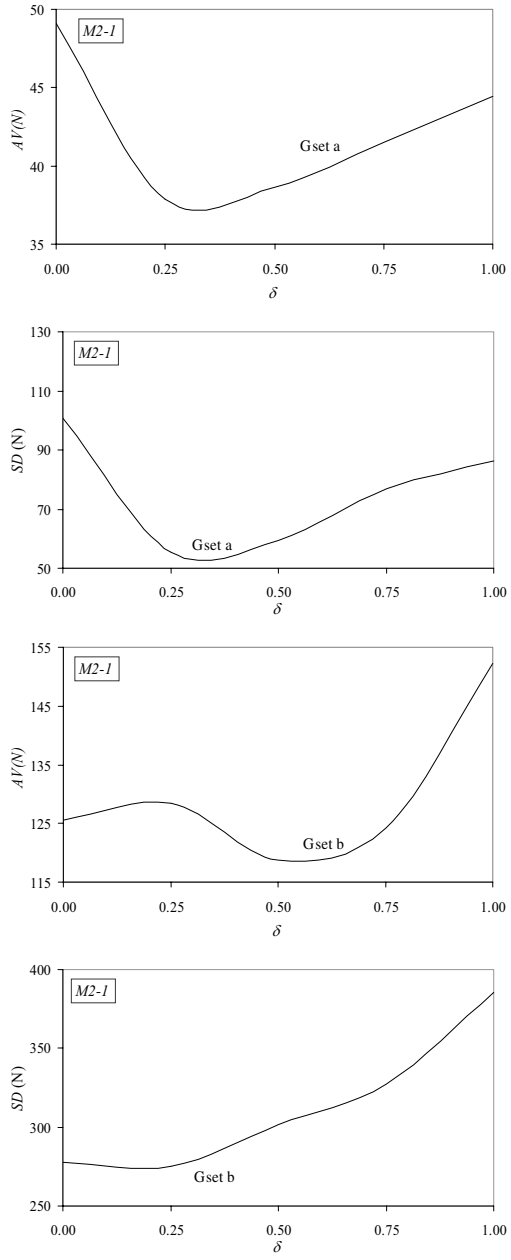


Figure 3: *M2-1* average number of generations to achieve a solution $AV(N)$ and standard deviation $SD(N)$ versus $\delta = \{0.0, 0.25, 0.5, 0.75, 1.0\}$ with Gsets a and b and F_s.

Due to the huge number of possible combinations of the GA parameters, in the sequel we evaluate only a limited set of cases. Therefore, *a priori*, other values can lead to different results. Nevertheless, the authors developed an extensive number of numerical experiments and concluded that the following cases are representative.

3.1 Using the static fitness function

In this sub-section we analyze the GA improvement when adopting a static fitness function including the discontinuity measure δ error.

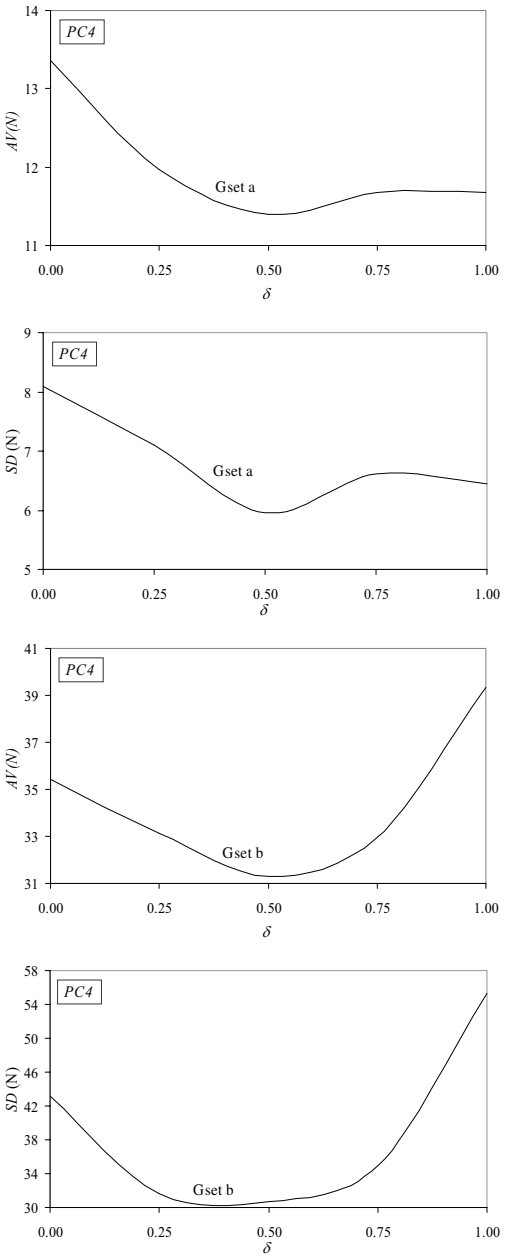


Figure 4: *PC4* average number of generations to achieve a solution $AV(N)$ and standard deviation $SD(N)$ versus $\delta = \{0.0, 0.25, 0.5, 0.75, 1.0\}$ with Gsets a and b and F_s.

Figures 3 and 4 show the average number of generations to achieve the solution $AV(N)$ and the corresponding standard deviation $SD(N)$ versus the discontinuity factor $\delta = \{0.0, 0.25, 0.5, 0.75, 1.0\}$, using Gset a and Gset b, for the *M2-1* and the *PC4* circuits, respectively.

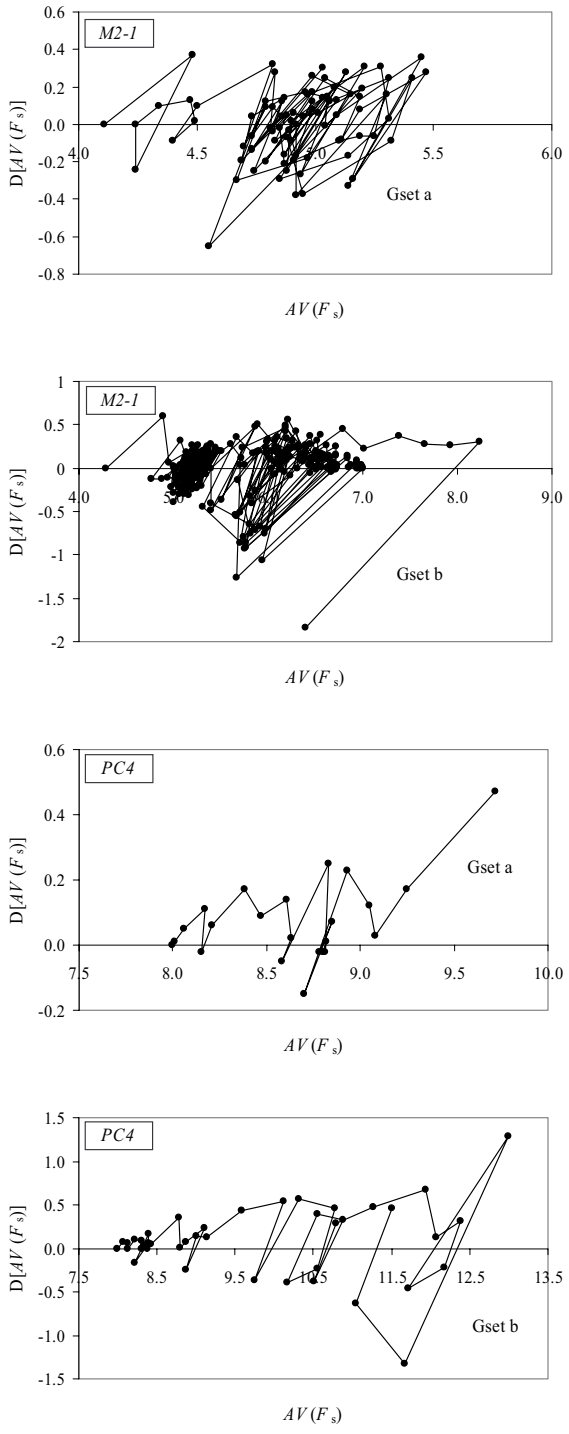


Figure 5: Phase plane of one GA run for the *M2-1* and the *PC4* circuits using Gsets a and b ($\delta = 0$), with F_s.

The results reveal that, as expected from previous studies [Reis et al., 2004], the RISC-like set Gset a presents better performance than the CISC-like gate set Gset b for all values of δ. On the other hand, analysing the influence of δ we conclude that the GA response is best mostly in the region around $\delta = 0.5$ for the two circuits and for the two gate sets.

Figure 5 depict the phase plane charts of the average of the static fitness function with $\delta = 0$ for the two circuits and the two gate sets.

Due to the stochastic nature of the GA evolution, the phase plane varies between experiments and generalization is not possible. Nevertheless, the charts indicate the global dynamics in each case.

3.2 Experiments using dynamic fitness function

In this sub-section, we analyze the GA performance when we adopt a dynamic scheme for the fitness function.

The first set of simulations investigate separately the differential scheme ($\mu = \{0.0, 0.25, 0.5, 0.75, 1.0\}$) and the integral scheme ($\lambda = \{0.0, 0.25, 0.5, 0.75, 1.0\}$) in F_d for gains $10^{-3} \leq K_D \leq 10^2$ and $10^{-3} \leq K_I \leq 10^2$, respectively.

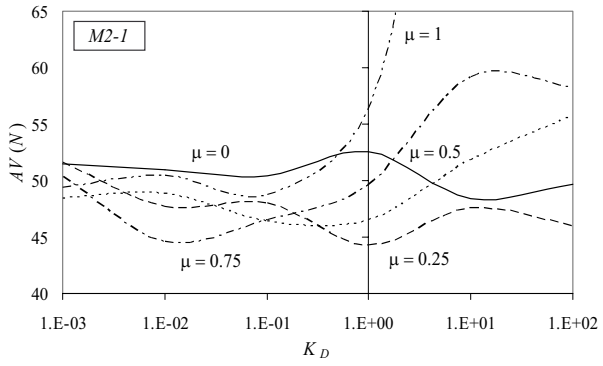
Figures 6-9 show the average number of generations to achieve a solution $AV(N)$ and the standard deviation $SD(N)$ for the differential PD^{μ} (*i.e.*, $K_I = 0.0$) and the integral schemes PI^{λ} (*i.e.*, $K_D = 0.0$), for the *M2-1* and *PC4* circuits, using the Gset a and the Gset b, respectively. The charts include the plots for $\mu = 0.0$ and $\lambda = 0.0$, that is without dynamic fitness, in order to ease the comparison.

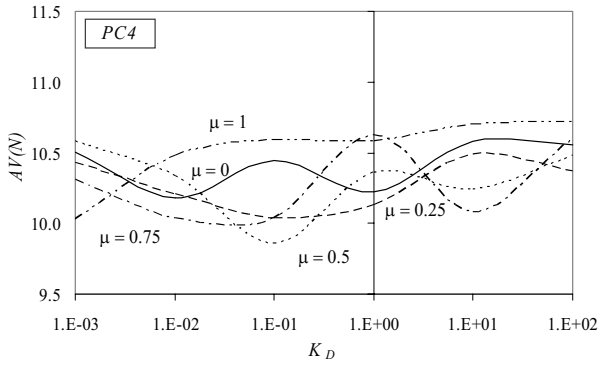
Since we achieved better results for $\mu = 0.25$ and $\lambda = 0.25$, we decided to investigate the combination of these parameters. Therefore, the second set of simulations evaluates the proportional-integral-differential $PI^{\lambda}D^{\mu}$ scheme. Due to the large number of possible combinations of $\{\lambda, \mu, K_I, K_D\}$ we establish $\lambda = \mu = 0.25$ and $10^{-3} \leq K_D = K_I \leq 10^2$.

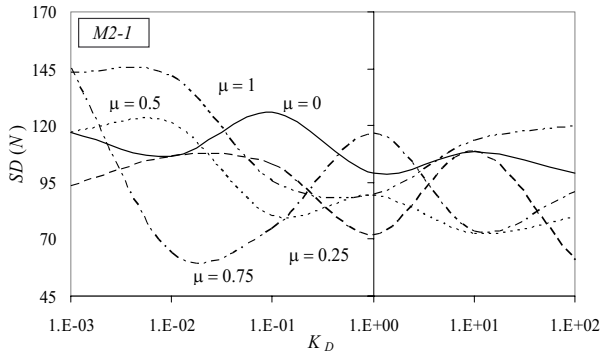
Figures 10-11 show the average number of generations to achieve a solution $AV(N)$ and the standard deviation $SD(N)$ for the proportional-integral-differential $PI^{\lambda}D^{\mu}$ scheme, for the *M2-1* and the *PC4* circuits, using Gset a and Gset b ($\delta = 0$), respectively.

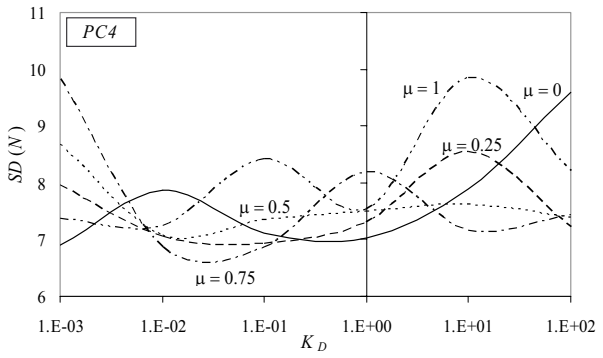
Comparing the previous $PD^{1/4}$ and $PI^{1/4}$ schemes with the $PI^{1/4}D^{1/4}$ case, we verify that the inclusion of both actions improves slightly the results.

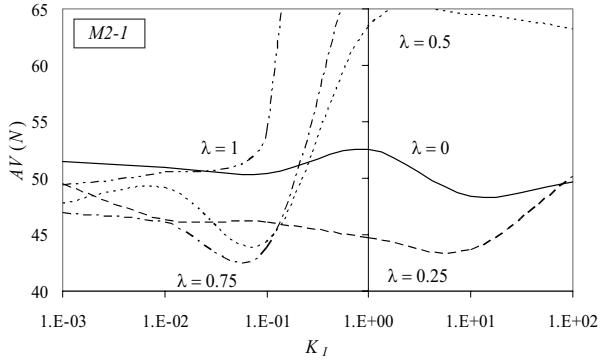
We conclude that the F_d concept produces better results than the classical F_s. Moreover, the results reveal that, the RISC-like Gset a presents a superior performance for all values of (λ, μ, K_I, K_D).

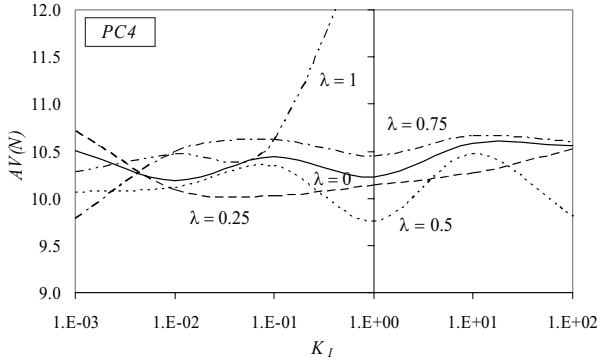


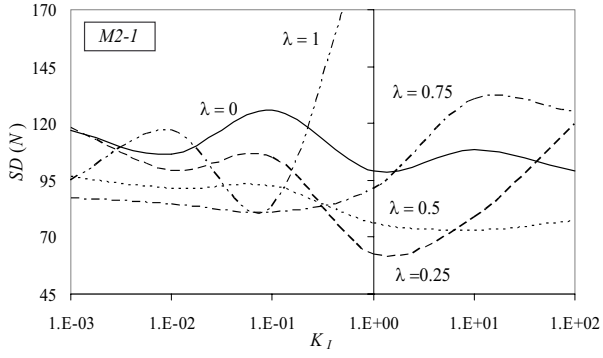












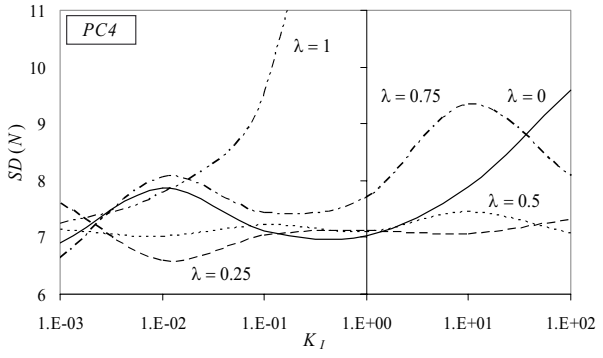


Figure 6: *M2-1* average number of generations to achieve a solution $AV(N)$ and standard deviation $SD(N)$ for the PD^{μ} and PI^{λ} schemes ($\delta = 0$) with Gset a.

Figure 7: *PC4* average number of generations to achieve a solution $AV(N)$ and standard deviation $SD(N)$ for the PD^{μ} and PI^{λ} schemes ($\delta = 0$) with Gset a.

In a third set of simulations, we include the error discontinuity measure in the $PI^{1/4}D^{1/4}$ scheme (figures 12 and 13).

Figure 14 shows several phase plane charts that occur for the $PI^{1/4}D^{1/4}$ with $K = K_D = K_I = 1$ and $\delta = 0.50$.

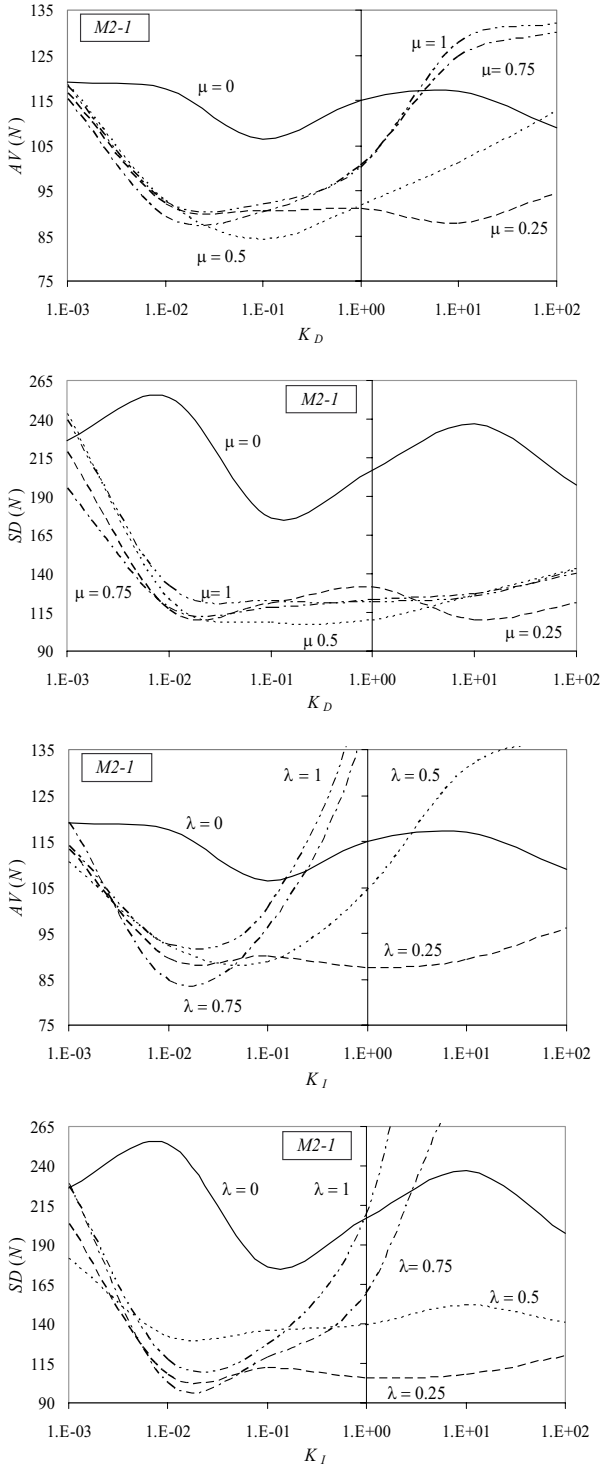


Figure 8: *M2-1* average number of generations to achieve a solution $AV(N)$ and standard deviation $SD(N)$ for the PD^{μ} and PI^{λ} schemes ($\delta = 0$) with Gset b.

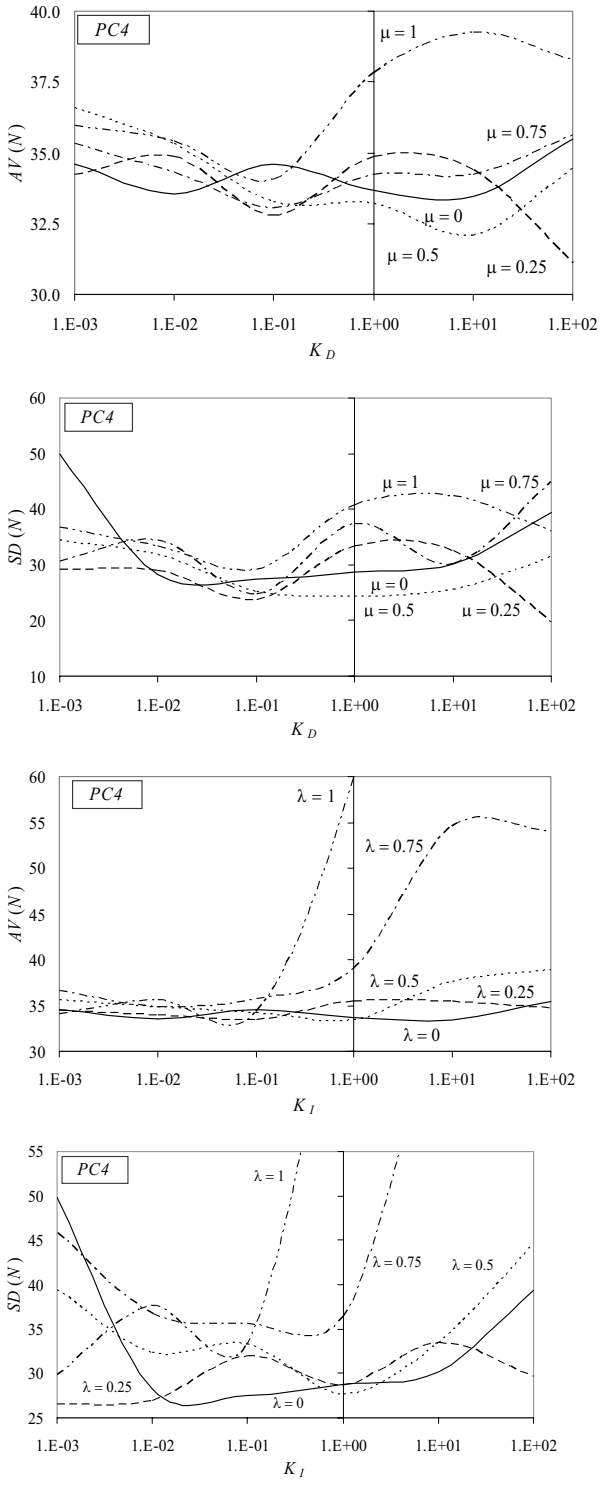
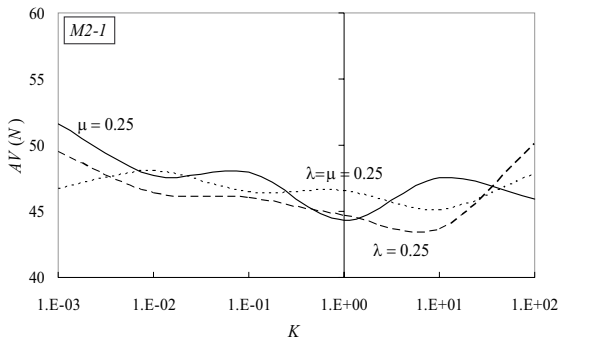


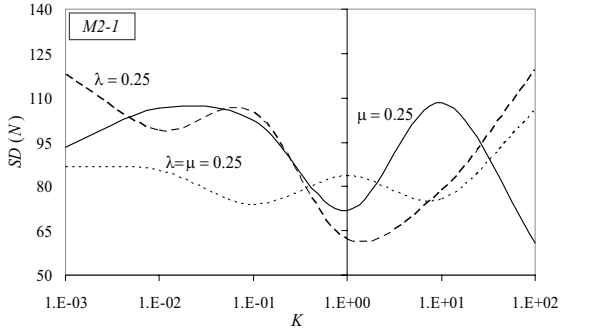
Figure 9: *PC4* average number of generations to achieve a solution $AV(N)$ and standard deviation $SD(N)$ for the PD^{μ} and PI^{λ} schemes ($\delta = 0$) with Gset b.

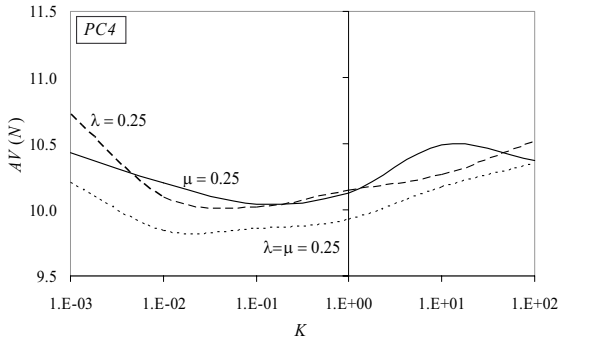
In conclusion, the introduction of discontinuity and dynamic effects improves significantly the GA performance.

4 Conclusions

This paper presented two techniques for improving the GA performance. Firstly, we concluded that we get superior results by measuring the error discontinuity.







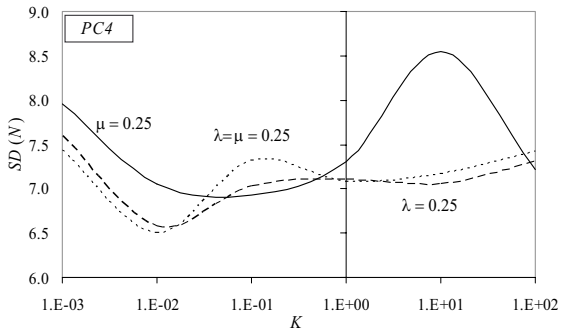
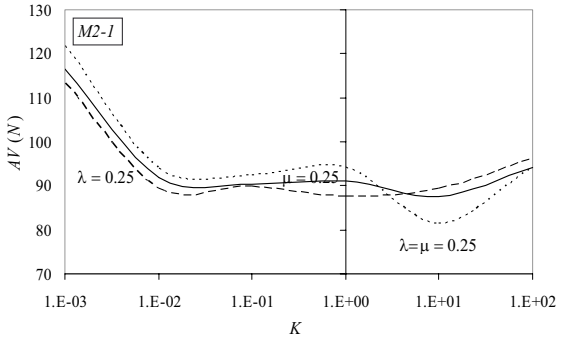
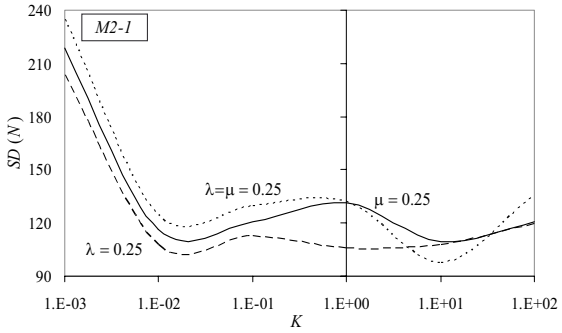
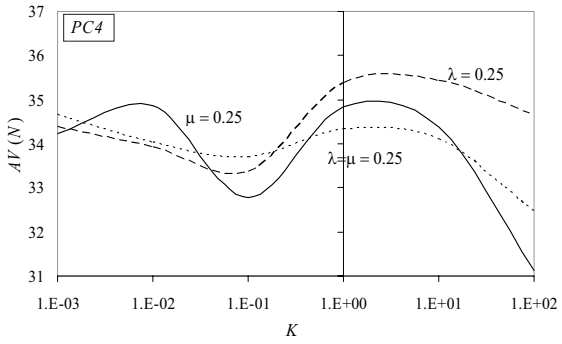


Figure 10: *M2-1* and *PC4* average number of generations to achieve a solution $AV(N)$ and standard deviation $SD(N)$ for the $PI^{\lambda}D^{\mu}$ scheme, for $K = K_D = K_I$, ($\delta = 0$) with Gset a.

Secondly, we verified that, the new concept of fractional-order dynamic fitness function constitutes an important method to outperform the classical static fitness function approach. The tuning of the 'optimal' parameters (λ, μ, K_I, K_D) was established by numerical evaluation.







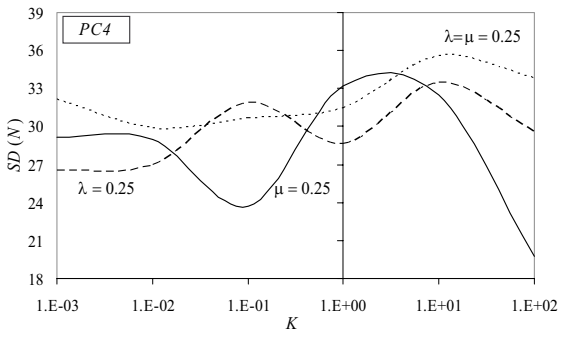
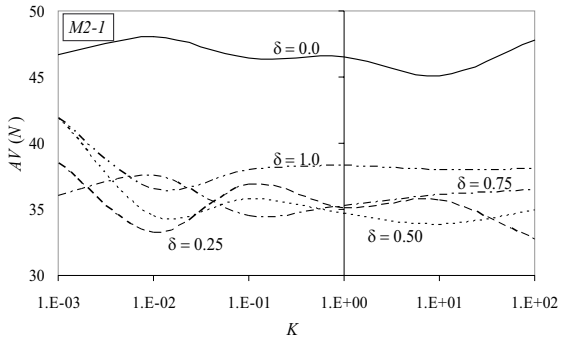
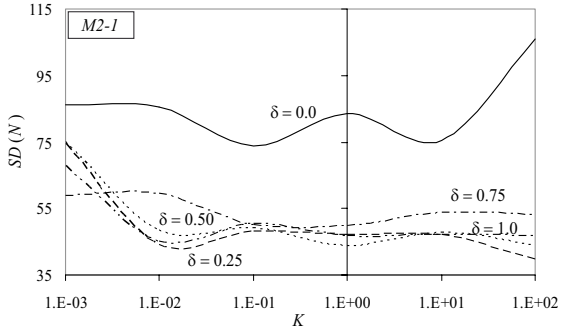
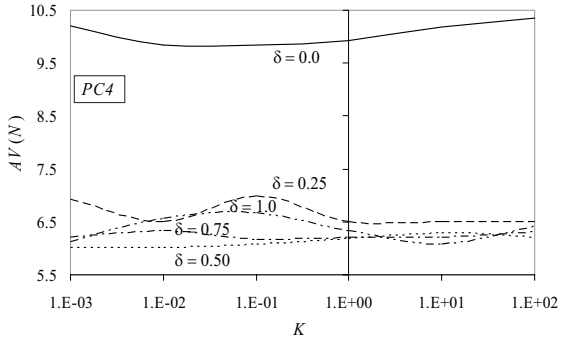


Figure 11: *M2-1* and *PC4* average number of generations to achieve a solution $AV(N)$ and standard deviation $SD(N)$ for the $PI^{\lambda}D^{\mu}$ scheme, for $K = K_D = K_I$, ($\delta = 0$) with Gset b.

Therefore, future research will address the problem of having a more systematic design method. Furthermore, these conclusions encourage further studies using fractional order dynamical schemes.







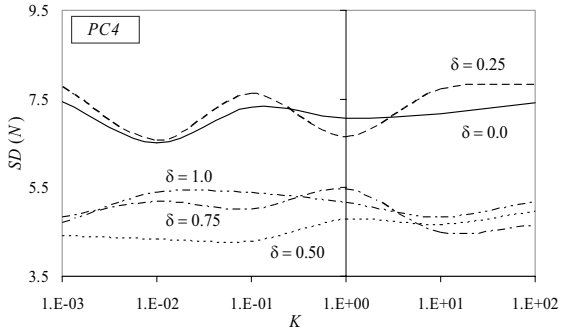
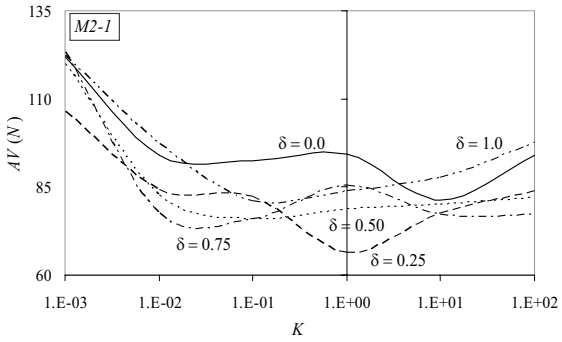
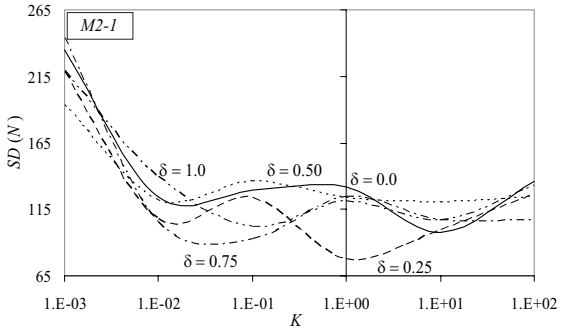
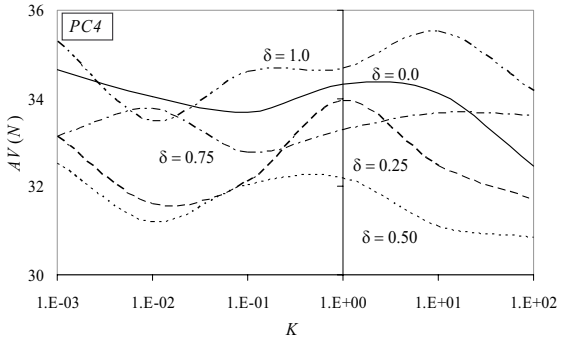


Figure 12: *M2-1* and *PC4* average number of generations to achieve a solution $AV(N)$ and standard deviation $SD(N)$ for the $PI^{1/4}D^{1/4}$ *versus* $K = K_D = K_I$ for $\delta = \{0.0, 0.25, 0.5, 0.75, 1.0\}$ with Gset a.







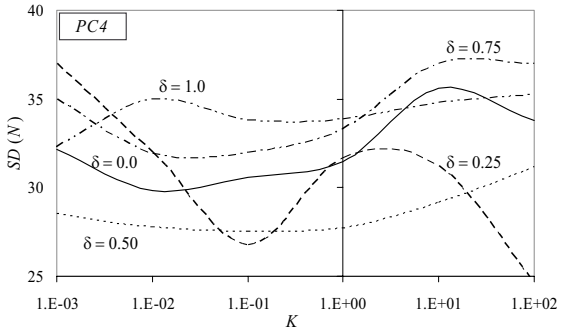


Figure 13: *M2-1* and *PC4* average number of generations to achieve a solution $AV(N)$ and standard deviation $SD(N)$ for the $PI^{1/4}D^{1/4}$ *versus* $K = K_D = K_I$ for $\delta = \{0.0, 0.25, 0.5, 0.75, 1.0\}$ with Gset b.

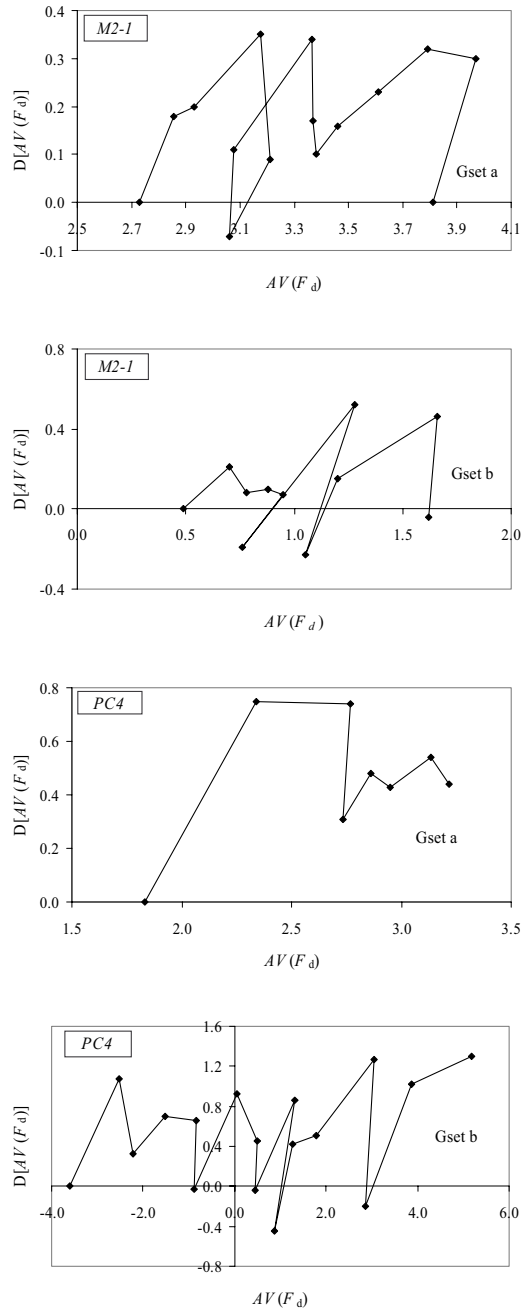


Figure 14: Phase plane for one GA run for the *M2-1* and the *PC4* circuits for the $PI^{1/4}D^{1/4}$ with $K = K_D = K_I = 1$ and $\delta = 0.50$ using Gsets a and b.

References

Coello, C., Christiansen, A. and Aguirre, A. (1996). Using Genetic Algorithms to Design Combinational Logic Circuits. *Intelligent Engineering through Artificial Neural Networks*. Vol. 6, pp. 391-396.

Gordon, T. and Bentley, P. (2002). Towards Development in Evolvable Hardware. In *Proc. of the 2002 NASA/DOD Conference on Evolvable Hardware*. pp. 241-250.

Kalganova, T., Miller, J. and Lipnitskaya, N. (1998). Multiple_Valued Combinational Circuits Synthesised using Evolvable Hardware. In *Proc. of the 7th Workshop on Post-Binary Ultra Large Scale Integration Systems*.

Koza, J. (1992). *Genetic Programming. On the Programming of Computers by means of Natural Selection*. MIT Press.

Louis, S. and Rawlins, G. (1991). Designer Genetic Algorithms: Genetic Algorithms in Structure Design. In *Proc. of the Fourth Int. Conference on Genetic Algorithms.*

Machado, J., (1997). Analysis and Design of Fractional-Order Digital Control Systems. *SAMS Journal Systems Analysis, Modelling, Simulation*, vol. 27: 107−122.

Méhauté, A. (1991). *Fractal Geometries: Theory and Applications*. Penton Press, London.

Miller, J., Thompson, P. and Fogarty, T. (1997). *Algorithms and Evolution Strategies in Engineering and Computer Science: Recent Advancements and Industrial Applications*. Wiley.

Miller, K. and Ross B., (1993). *An Introduction to the Fractional Calculus and Fractional Differential Equations*. John Wiley & Sons, New York.

Oldham, K. and Spanier, J. (1974). *The Fractional Calculus: Theory and Application of Differentiation and Integration to Arbitrary Order*. Academic Press, New York.

Oustaloup, A., (1995). La *Dérivation Non Entier: Théorie, Synthèse et Applications*. Ed. Hermes.

Reis, C., Machado, J. and Cunha, J., (2004). Evolutionary Design of Combinational Logic Circuits. *Journal of Advanced Computational Intelligence and Intelligent Informatics*. Fuji Technology Press, vol. 8, No. 5, pp. 507-513.

Sano, Y and Kita, H. (2000). Optimization of Noisy Fitness Functions by means of Genetic Algorithms using History of Search. In *Proc. of PPSN VI*, pp. 571-581.

Thompson, A. and Layzell, P. (1999). Analysis of unconventional evolved electronics. *Communications of the ACM*, vol. 42, pp. 71-79.

Torresen, J. (1998). A Divide-and-Conquer Approach to Evolvable Hardware. In *Proc. of the Second International Conference on Evolvable Hardware*. Vol. 1478, pp. 57-65.

Vassilev, V. K. and Miller, J. F. (2000). Scalability Problems of Digital Circuit Evolution. In *Proc. of the Second NASA/DOD Workshop on Evolvable Hardware*. pp. 55-64.

Westerlund, S., (2002). *Dead Matter Has Memory! Causual Consulting*. Sweden: Kalmar.

Zebulum, R. S., Pacheco, M. A. and Vellasco, M. M. (2001). *Evolutionary Electronics: Automatic Design of Electronic Circuits and Systems by Genetic Algorithms*. CRC Press.