International Journal of Applied Engineering Research ISSN 0973-4562 Volume 12, Number 8 (2017) pp. 1469-1477

Design and Implementation of a Controller Using Genetic Algorithms

Juan M. Chaparro F.
Professor, Department of Electronics Engineering, Central University, Bogotá, Colombia.

Oscar F. Aviles S.
Professor, Department of Electronics Engineering, Central University, Bogotá, Colombia.

Alonso J. Chica L
Professor, Department of Electronics Engineering, Central University, Bogotá, Colombia.

Abstract
This work presents the implementation of a control application through the use of genetic algorithms in a motorized pendulum. The dynamic model of a motorized pendulum is presented and operating criteria are specified, the implementation in Matlab is performed for the respective validation of the operation of the control system.

Keywords: Evolution, crossover, mutation, genetic, controller.

INTRODUCTION

Evolutionary Computing was introduced in the high sixties by I. Rechenberg in his work "Evolutionary Strategies". This idea was developed by other researchers, until in 1975 John Holland and his colleagues published the book "Adaption in Natural and Artificial System", as an analogy to the genetic coding of living beings, who stored their physical characteristics in genes Which are arranged linearly to form chromosomes, optimization problems using genetic algorithms store their minimal units of information for each element of the population in genes, this finite-length string forms the chromosomes in turn.

The traditional coding proposed by Holland frames chains of chromosomes formed by zeros and ones but a large number of symbol patterns can be used for a representation. Many problems have complex objective functions and optimization tends to end in local minimums or maximums. The idea of genetic algorithms is to optimize an objective function using the principles of natural selection on the parameters of the function.

GENETICS ALGORITHM

Genetic algorithms - GA are inspired by the problem solving methods that nature has used to make wines evolve, adapting them to different environments. The technique used by nature consists mainly in the coding of the characteristics of wines in the genome, and their evolution through sexual reproduction and mutations. This basic idea, which allows wine beings to adapt to the environment in which they live, and which is nothing other than their optimization for given boundary conditions, encourages the use of algorithms based on these techniques as general methodology of optimization Systems [1].

GAs are based on the principle of evolution: survival of the fittest for adaptation to the environment. They use biological analogy to develop a population of individuals for generations towards an optimal solution. The evaluation of the solutions obtained is performed by means of a function called objective. They look for a population of points in parallel and do not require derivative information. Use transition rules Probabilistic variables. They are usually more direct to apply. They can give several potential solutions to a problem and are therefore more robust than traditional methods.

Components of GA

The components of a genetic algorithm [3] are:

- A function to optimize.
- A group of candidates for solution
- An evaluation function that measures how candidates optimize the function.
- Playback function.

The genetic algorithm works because there is competition for resources and inherited the best genetic configuration in each generation. Certain characteristics of an organism make it more successful than other organisms of the same species, where success is defined as reproductive success, the better adapted leaves more children than the others. Heritable traits that favor success tend to be more represented in the population.

The evolution occurs due to two processes:

- Natural selection: Determines which members of the population will survive to reproduce. The process of natural selection is a process of normalization, the variation present at any given time in the population is assumed to appear at random.
- Sexual reproduction: Ensures the mixing of genes among children. The mode of reproduction that produces the greatest variance is the sexual one, this mixes genes of different individuals to form the new generation. Asexual reproduction only produces variation through external perturbations such as mutation.

The genetic algorithm initially creates a population of chromosomes, each chromosome is a string of 0, 1, *, where the "*" symbol means no matter. The sections of each chromosome are called genes and represent the unknown parameters that will

be encountered with the genetic algorithm to optimize an objective function J.
After evaluating the objective function for each chromosome, parents of the next generation are chosen by a selection method. After selecting the parents, the couples are randomly formed and reproduced with the crossover operator. The cross-breeding site is chosen at random, the binary chains are crossed simply by chaining one after the other.

GA Characteristics

GAs are search algorithms based on the mechanics of natural selection, where the best survive. Some characteristics of GAs are:

- They work with a code corresponding to the parameters, that is, the parameters are coded as a binary string of finite length.
- They allow a quick search in the parameter space and avoid local minimums or mimes.
- Exploit the objective function information, no gradient calculations are required.
- You have a population of possible solutions not a single solution. When it finds one, it is removed from the population (penalizing the objective function for that solution) and continuous optimization
- They are fast and robust.

Search process of GA

The search process for a genetic algorithm [2] is seen in Figure 1:

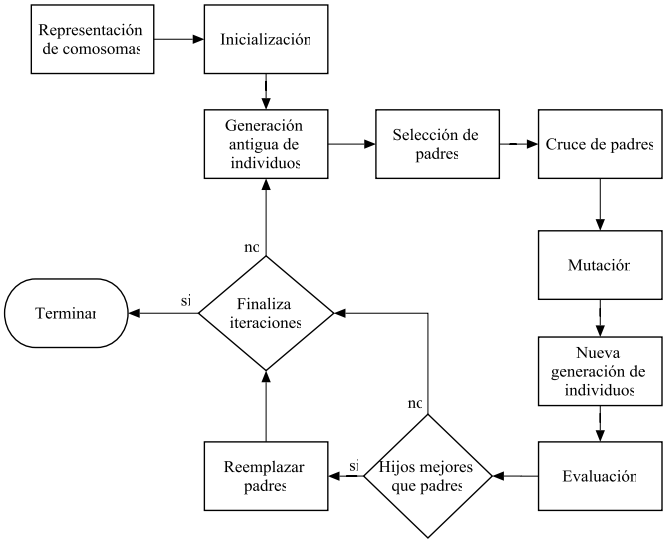


Figure 1. GA Search Process

The optimization process begins with initialization, consisting of the generation of a population of individuals. There are two interpretations for the concept of individual and population in the optimization of controllers.
The first, commonly called Pitts, which is the most used, understands by population a group of individuals; In this article, each individual will be a complete controller. In this context, the following definitions are made:

- Genome: All the parameters that define each and every individual of the population.
- Genotype: The part of the genome that describes a particular individual. In this interpretation are the genes Which describe a particular controller.
- Phenotype: The expression of a genotype. In this interpretation it is a concrete controller
- Gen: Each of the parameters that describe an individual. In this interpretation 108 genes encode parameters of a particular controller.

The second interpretation, usually called Michigan, consists in considering as individual the abilities of a controller, so that the population represents the set of all its abilities. According to this interpretation, the genome is the coding of these abilities. This type of interpretation has been used in real-time systems control and robotics.

For example, in the case of robotics, an individual is formed by the set of rules that allow the robot to grasp an object; The population consists of individuals specializing in certain tasks, and during evolution the best ones are selected for each task.

During the initialization phase a population can be generated, either randomly, or from an existing controller. The latter allows starting solutions developed by other procedures, both automatic and manual.

The next phase of the process is the evaluation, in which it is given that each of the controllers that form the population acts to control the system, usually through a simulation, being evaluated by means of an efficiency function. This is usually the stage that requires the most time, since each of the population controllers must be simulated for the time necessary to evaluate their efficiency, and in a number of sufficient control situations. The definition of this function of efficiency is fundamental in the success of the use of GA for the resolution of a given problem. From it, the designer decides which behaviors are not to be strengthened, which ones, and to what extent.

The next phase is the selection phase, in which the process of natural selection of individuals in each generation is simulated. In this sense, individuals must be selected to transmit their genotype to the next generation. Given a population of M individuals, several alternatives are possible:

- Only the best is selected. From the entire population the individual is chosen who has a better evaluation, and only his genome is used to create the next generation.
- Only the best are selected. From the entire population we choose the n individuals (n << M) who have a better evaluation, and to create the next generation only their genome is used.
- All can be selected. Each individual of the population is selected with a higher probability the better his evaluation.

The selection phase is followed by the application of a sequence of genetic operators that simulate the process of sexual reproduction of wine beings. By applying these operators to the genome that has been selected, a final genome is obtained for the next generation. This phase is then defined by the sequence of operators selected, which also serves to identify the genetic algorithm used.

With the final genome, corresponding to the following generations, a phenotype is expressed, reconstructing each one of the controllers that form the population and proceeding again to its evaluation. This process is repeated a fixed number of times, or until the evaluation stabilizes.

The most commonly used method for coding genes is the use of a fixed length and sequence string. The parameters of the controller that has been decided to adjust according to the criteria that have been exposed are stored in this string as fixed sequences of numbers. In this way, the number of genes (in our case parameters) is fixed, and its value is always stored in the same position of the chain.

On the other hand, we also have to decide how to code these numbers; 8-bit and 16-bit binary encodings are common, although fixed or floating-point coding may also be used. Also, a coding based on reserving a certain number of bits for each gene may be used, and encoding their value by the number of "1" present, regardless of their position. The type of coding selected for the genes will influence how operators are implemented (such as the mutation) as well as the amount of memory needed to store the genome.

Operators used in genetic algorithms make modifications on the genome, simulating the evolution of the genome of wine beings caused by sexual reproduction and other factors, such as mutations. The main operators used in genetic algorithms are crossbreeding and mutation.

Crossing allows to simulate the genome's combination between individuals. This can be done on pairs of individuals or on the whole population. When it is performed on pairs of individuals (parents), the genome is copied into two memories, randomly selecting a point in this memory, and crossing the two halves of each copy of their genotypes.

When performed on the whole population, the cross-breeding can be performed simultaneously on the whole genome by the procedure known as Russian roulette: first the genome is copied into two circular memories, then one of them rotates a random angle, then Sc In this memory a point of form also random and, finally, the two halves of each copy of the genome are crossed.

By executing one way or another, crossbreeding allows the best qualities of the parents to be combined in their offspring. In problems in which the controller is expected to simultaneously achieve several objectives, it is expected that descendants at some point in evolution can inherit from each of their parents the set of rules that allow each of the objectives to be achieved.

However, it is known that during the reproduction of wines, errors occur in the copy of the genome, which are assigned a certain probability p. This effect produces diversity in the genome, and allows exploring new solutions to problems. This is modeled by the mutation operator, which involves selecting a mutation point and modifying the stored parameter with a probability p. If the encoding is a binary type, one of the bits is modified, whereas if the encoding is fixed or floating type, a random change is usually made within a limited range.

Inverted Pendulum motorized

We propose to perform a controller using genetic algorithms. The controller is designed for a motorized pendulum, figure 2.

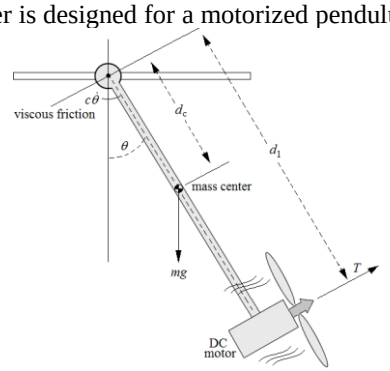


Figure 2. Plant problem

The equation of the motorized pendulum is the following:

$$J\ddot{\theta} = Td_1 - c\dot{\theta} - mgd_c\text{sen}\theta \qquad (1)$$

Where:

T : torque
J : momento of inertia
θ : Angular position
m : mass
g : gravity

Considering (1) the implementation in Simulink is realized as seen in the figure 3:

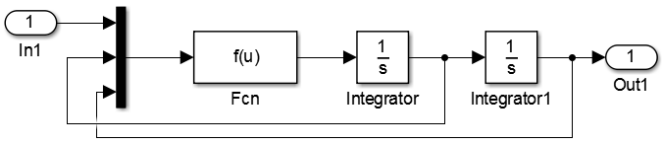


Figure 3. Implementation of the plant in simulink.

Where `fcn` corresponds to $\ddot{\theta}$, figura 4.

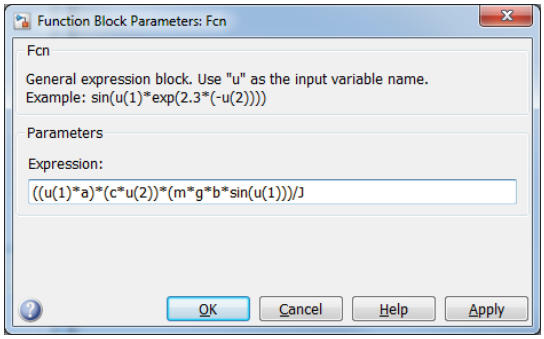


Figure 4. Equation de `fcn`

Because a controller must be implemented with GA [4], the overall structure of a control system can not be lost sight of. Therefore, a control system was implemented using Simulink and the block representing the plant to acquire random data assuming random values for the controller parameters. This is shown in the following figure 5.

The sampling time set was 0.1s to have a common parameter for all blocks because it is a digital control system. This value was obtained from trial and error.

The equation of the digital controller is proposed as follows:

$$C(q) = \frac{a_0 + a_1 q + a_2 q^2}{1 + b_1 q + b_2 q^2} \qquad (2)$$

From the simulations made from the variation of the parameters of the controller and observing the values of the system error and the output of the controller, Table 1 was obtained.

Table 1. Parameters Variation

a2	a1	a0	b2	b1	Error	Manipulada
3.5	2.5	0	2.5	3.5	3.327	3.189
3	2	0	2	3	3.423	3.353
3.5	2.5	1	2.5	3.5	2.852	3.504
1	2	0	1	0	1.905	3.233
0.5	0.1	0.3	1	2	2.278	40.32
3	2	1	2	1	1.903	2.851
0.5	0.5	1	2.5	3.5	9.916	3.184
2	1	2	2	1	2.272	2.853
1.1	1.5	1	2.5	1.5	3.963	2.851
2	3	2	1	1	1.218	3.092
3.1	3.5	4	3.5	1.5	1.598	2.991
1	1	1	1	1	2.852	2.852
2	2	2	2	2	2.376	2.854
2.5	2.8	2.2	1	0	0.8431	3.004
3	5	0	2	1	1.426	2.852

This data table and the values given to $[a_2, a_1, a_0, b_2, b_1]$ corresponds to the initial population that in this case is 15.

GA Implementation

From Table 1, we selected the best products that had a fairly close knowledge of the value that the controller had to deliver to the plant from the Simulink simulation. The best 8 individuals were taken as shown in Table 2.

Table 2. Best Parameters

a2	a1	a0	b2	b1	Error	Manipulada
1	2	0	1	0	1.905	3.233
0.5	0.1	0.3	1	2	2.278	40.32
3	2	1	2	1	1.903	2.851
2	3	2	1	1	1.218	3.092
3.1	3.5	4	3.5	1.5	1.598	2.991
2	2	2	2	2	2.376	2.854
2.5	2.8	2.2	1	0	0.8431	3.004
3	5	0	2	1	1.426	2.852

These individuals are configured in the parents of the next generation.

With these parents, we proceed to perform the different steps of crossing, mutation, improvement of the population and stop the algorithm.

The program performed in Matlab® is explained below. With the selected parents $[a_2, a_1, a_0, b_2, b_1]$, proceed to choose a random pair that is performed with the rand function. Since the location of the parents is from 1 to 8, the random number is rounded to the nearest whole number.

Algorithm 1:

father1= *rand*(1) * 7 + 1
father1=*round(*father1)
father2 - *rand*(1) * 7 + 1
father2 - *round(*father2*)*

Obtaining the parents proceed to perform the procedure of crossing. This crossing was performed by exchanging the values of a_0, b_2 and b_1 of the padrel with those of the father2.

Algorithm 2:

*cruz*1*a*2=*a*2(father1)
*cruz*1*a*1=*a*1(father1)
*cruz*1*a*0=*a*0 (father1)
*cruz*1*b*2=*b*2(father1)
*cruz*1*b*1=*b*1(father1)
*cruz*2*a*2=*a*2(father2)
*cruz*2*a*1=*a*1(father2)
*cruz*2*a*0=*a*0 (father2)
*cruz*2*b*2=*b2*(father2)
*cruz*2*b*1= *b*1(father2)
*w*1= *cruz*1*a*0
*w*2 =*cruz*1*b*2
*w*3= *cruz*1*b*1
*cruz*1*a*0=*cruz*2*a*0
*cruz*1*b*2=*cruz*2*b*2
*cruz*1*b*1=*cruz*2*b*1
*cruz*2*a*0=*w*1
*cruz*2*b*2=*w2*
*cruz*2*b*1=*w3*

The next step is to perform the mutation. The mutation consists of changing a bit randomly or child from the previous crosses. For this, it is necessary to perform a real-to-binary conversion and perform the mutation, which in this case was performed on the most significant bit for ease and simplicity.

Algorithm 3: Conversion to binary

```
bval1 = dec2bin(cruz1b1* 10, 8)
bval2 = dec2bin(cruz2b1 * 10, 8)
bval1 = bin2dec(bvall)
bval2 = bin2dec(bval2)
```

The mutation is made, finding the decimal value of the binary number. According to this value it is proceeded to establish conditions to change the most significant bit.

Since they are 8-bit data, if greater than 127, 128 is subtracted, and if e is less than 127, 128 is added to change the most significant bit.

Algorithm 4:

```
if bval1<127              ; From the first junction
   bval1= bval1+ 128
else
   bval1= bval1- 128
end
if bval2<127              ; From the second junction
   bval2= bval2+ 128
else
   bval2= bval2- 128
end
```

With the data mutated, the two new children are configured:

Algorithm 5:

```
cruz1b1=(bval1)/10    ; Reissuing the data
cruz2b1=(bval2)/ 10   ; Reissuing the data
a2son1=cruzla2
a1 son1=cruz1a1
a0 son1=cruz1a0
b2son1=cruz1b2
b1 son1=cruz1b1
a2 son2=cruz2a2
al son2=cruz2a1
a0 son2=cruz2a0
b2 son2=cruz2b2
b1 son2=cruz2b1
cruz1b1=(bval1)/10    ; Reissuing the data
cruz2b1=(bval2)/10    ; Reissuing the data
```

To verify whether these children are effectively used to enter the new population of individuals and to replace their parents, verification tests are necessary. The verification test for this case is, take a random data of the variable e (error) that is in the table of the original 15 individuals. The same happens with the variable m (manipulated). In addition, the controller C (q) can be represented as follows:

$$\frac{m(q)}{e(q)} = C(q) = \frac{a_0 + a_1 q + a_2 q^2}{1 + b_1 q + b_2 q^2} \tag{3}$$

In terms of z

$$\frac{M(z)}{E(z)} = C(z) = \frac{a_0 + a_1 z + a_2 z^2}{1 + b_1 z + b_2 z^2} \tag{4}$$

In terms of z^{-1}

$$\frac{M(z)}{E(z)} = C(z) = \frac{a_0 z^{-2} + a_1 z^{-1} + a_2}{z^{-2} + b_1 z^{-1} + b_2} \tag{5}$$

That in equation in differences is expressed as:

$$M(z)(z^{-2} + b_1 z^{-1} + b_2) = E(z)(a_0 z^{-2} + a_1 z^{-1} + a_2) \tag{6}$$

De donde

$$m(k) = \frac{a_0}{b_2} e(k-2) + \frac{a_1}{b_2} e(k-1) + \frac{a_2}{b_2} e(k) - \frac{b_1}{b_2} m(k-1) - \frac{1}{b_2} m(k-2) \tag{7}$$

Randomly choose the error variable and the variable Manipulated

Algorithm 6:

```
ent = rand(1)*14 + 1
ent = round(ent)
ent1 = rand(1) * 14 + 1
ent1= round(ent1)
ent2= rand(1) * 14 + 1
ent2= round(ent2)
man1=rand(1) * 14 +1
man1=round(man1)
man2=rand(1) * 14 +1
man2= round(man2)
```

Calculation of the manipulated variable for the first child

Algorithm 7:

$$m2 = \frac{a2son1}{b2son1} * e(ent) + \frac{a1son1}{b2son1} * e(ent1) + \frac{a0son1}{b2son1} * e(ent2) - \frac{b1son1}{b2son1} * m(man1) - \frac{1}{b2son1} * m(man2)$$

if $m2 < 60$; Acceptance condition and replaces parent

if $m1(\text{father1}) < m2$

$a2(\text{father1}) = cruz1a2$

$a1(\text{father1}) = cruz1a1$

$a0(\text{father1}) = cruz1a0$

$b2(\text{father1}) = cruz1b2$

$b1(\text{father1}) = cruz1b1$

$m1(\text{father1}) = m2$

end

end

Calculation of the manipulated variable for the second child

Algorithm 8:

$$m3 = \frac{a2son2}{b2son2} * e(ent1) + \frac{a1son2}{b2son2} * e(ent1) + \frac{a0son2}{b2son2} * e(ent2) - \frac{b1son2}{b2son2} * m(man1) - \frac{1}{b2son2} * m(man2)$$

if $m3 < 60$; Acceptance condition and replaces parent

if $m1(\text{father2}) < m3$

$a2(\text{father2}) = cruz2a2$

$a1(\text{father2}) = cruz2a1$

$a0(\text{father2}) = cruz2a0$

$b2(\text{father2}) = cruz2b2$

$b1(\text{father2}) = cruz2b1$

$m1(\text{father2}) = m3$

end

end

EVIDENCE AND DATA OBTAINED

From several tests and changes in the number of iterations, three possible drivers were obtained:

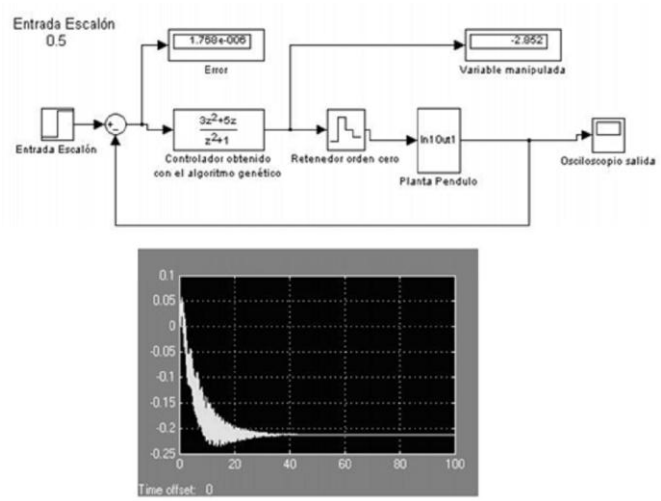
$$C(z) = \frac{5z + 3z^2}{1 + z^2} \quad (7)$$

Next figures show the behavior of the system with the first controller.

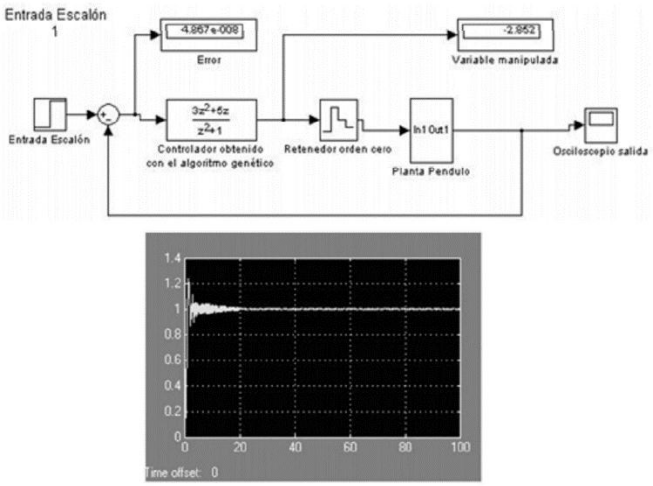
As seen in the previous figures of the first controller, there is an error in the system with all the inputs and there are quite strong oscillations in some cases.

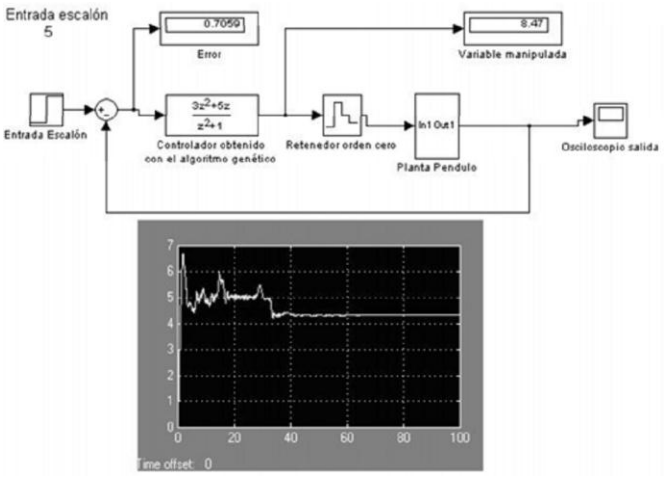
For the second controller:

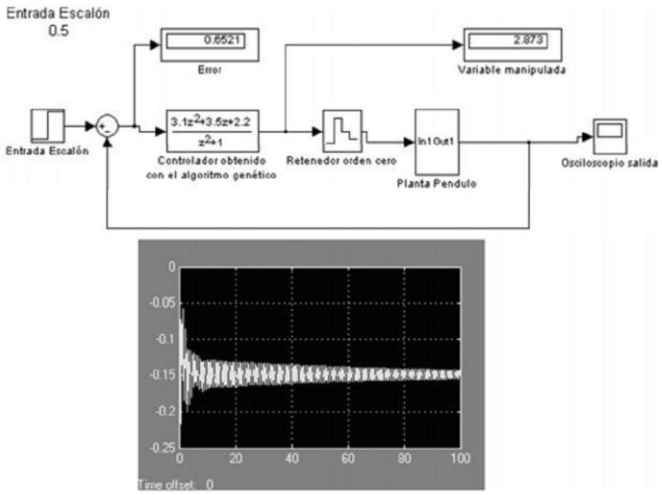
$$C(q) = \frac{3.1z^2 + 3.5z + 2.2}{z^2 + 1} \quad (8)$$

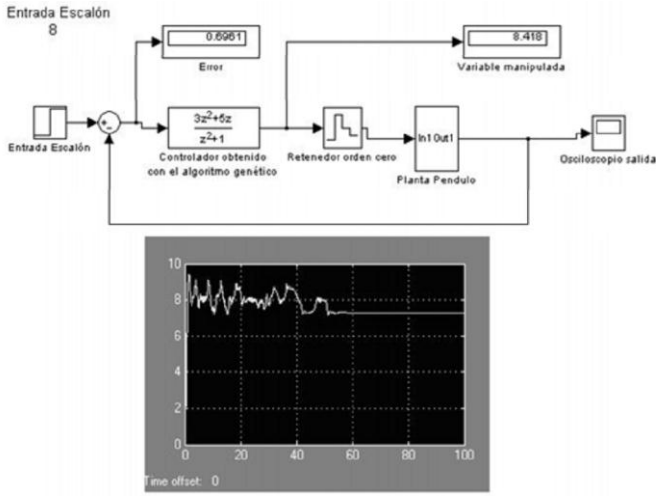


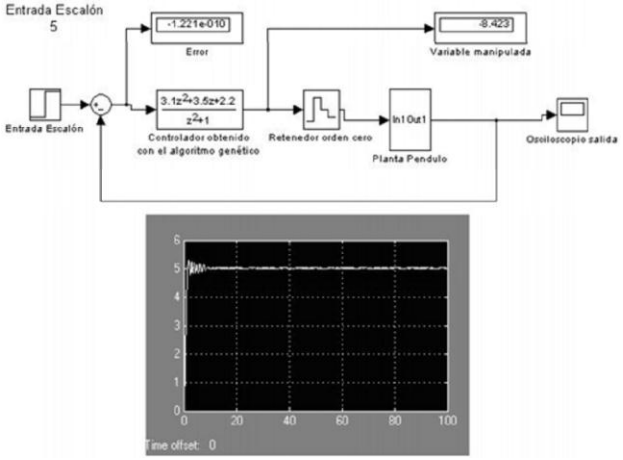
Response with scaling input 0.5 and first control

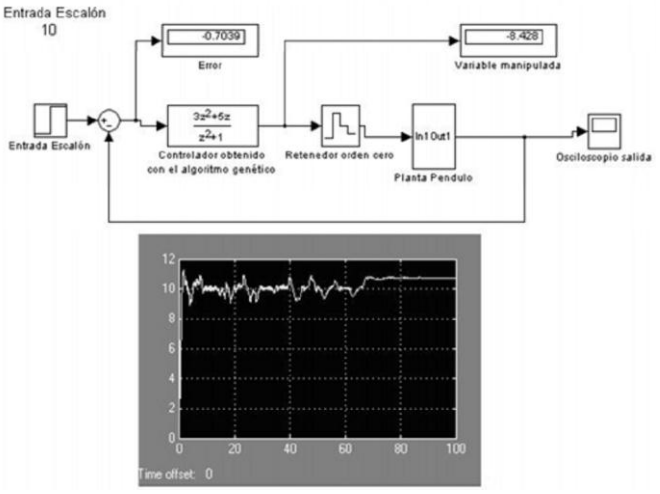


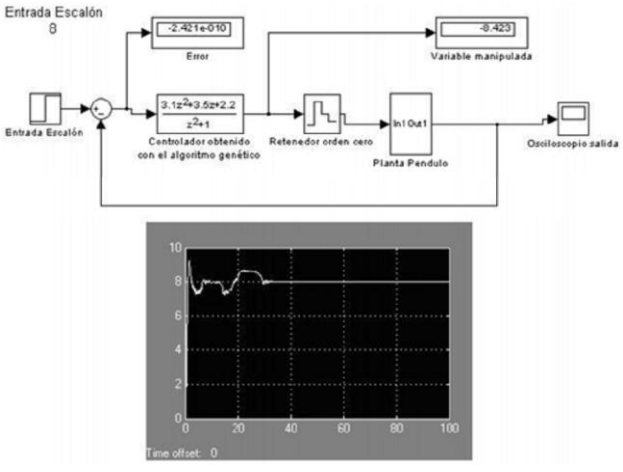

Entrada escalón
5
0.7059
Error
8.47
Variable manipulada
Entrada Escalón
Controlador obtenido
con el algoritmo genético
Retenedor orden cero
In1 Out1
Planta Pendulo
Osciloscopio salida
Time offset: 0

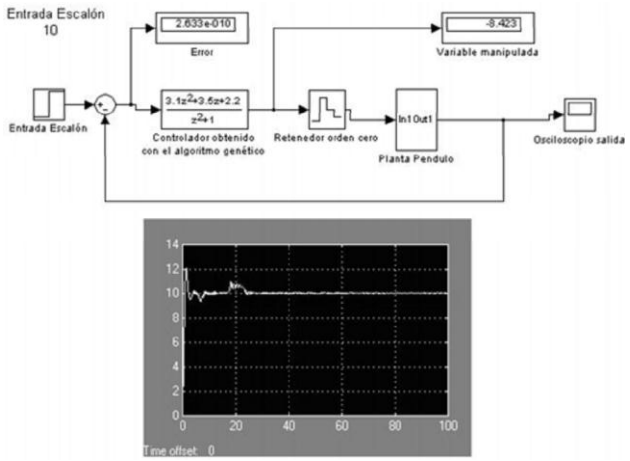

Entrada Escalón
0.5
0.6521
Error
2.873
Variable manipulada
Entrada Escalón
Controlador obtenido
con el algoritmo genético
Retenedor orden cero
In1 Out1
Planta Pendulo
Osciloscopio salida
Time offset: 0

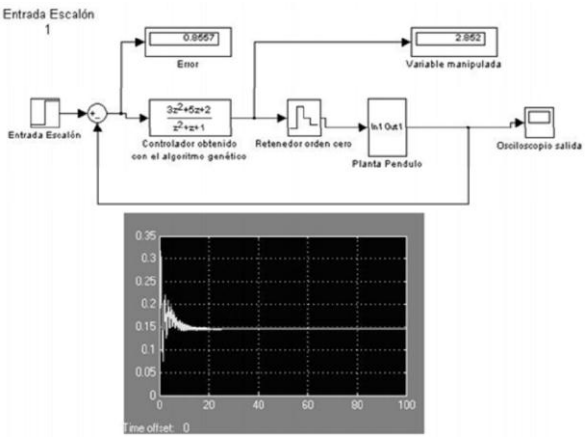

Entrada Escalón
8
0.6961
Error
8.418
Variable manipulada
Entrada Escalón
Controlador obtenido
con el algoritmo genético
Retenedor orden cero
In1 Out1
Planta Pendulo
Osciloscopio salida
Time offset: 0


Entrada Escalón
5
-1.221e-010
Error
-8.423
Variable manipulada
Entrada Escalón
Controlador obtenido
con el algoritmo genético
Retenedor orden cero
In1 Out1
Planta Pendulo
Osciloscopio salida
Time offset: 0


Entrada Escalón
10
-0.7039
Error
-8.428
Variable manipulada
Entrada Escalón
Controlador obtenido
con el algoritmo genético
Retenedor orden cero
In1 Out1
Planta Pendulo
Osciloscopio salida
Time offset: 0


Entrada Escalón
8
-2.421e-010
Error
-8.423
Variable manipulada
Entrada Escalón
Controlador obtenido
con el algoritmo genético
Retenedor orden cero
In1 Out1
Planta Pendulo
Osciloscopio salida
Time offset: 0





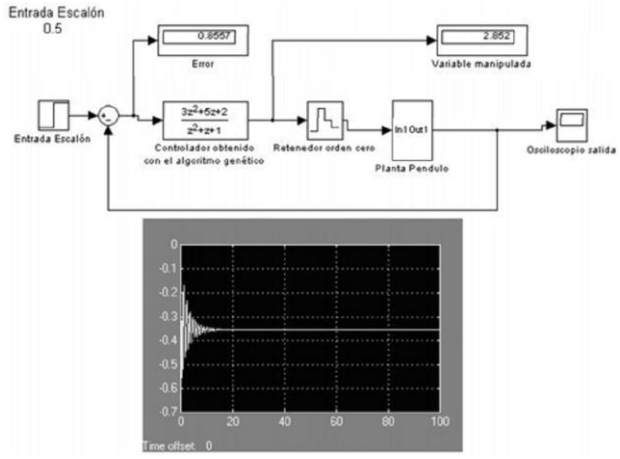
Next figures show the behavior of the system with the second controller.

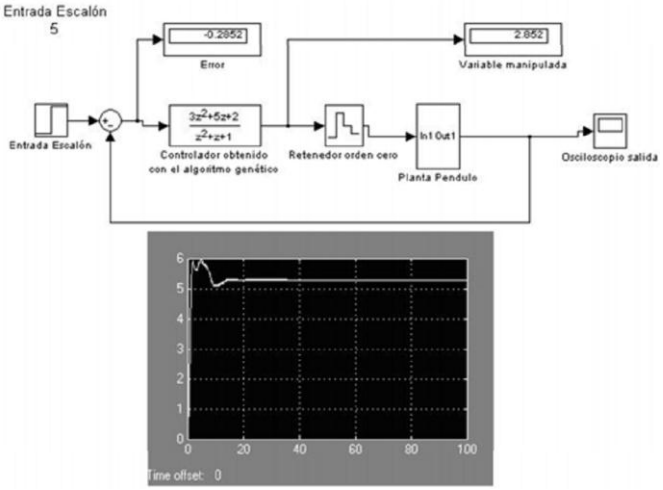
As observed in the previous figures of the second controller, it presents a better error behavior in the system reaching very small values with all inputs, except for the input 0.5, and presents oscillations, in most cases, in the First moments

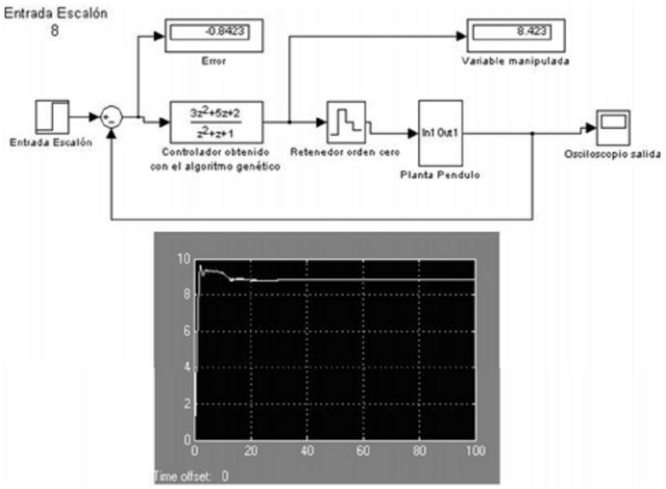
For the third controller:

$$C(q) = \frac{3z^2 + 5z + 2}{z^2 + z + 1} \tag{9}$$

Next figures show the behavior of the system with the third controller.







As shown in the previous figures of the third controller, it presents different variations of error in the system reaching high values in most of the inputs, and quite varied responses in oscillations not very suitable for the output of the system.

CONCLUSIONS

Genetic algorithms do not need much information. Only an objective function to evaluate the goodness of the solutions. They are easy to implement. They lack the constraints of other optimization methods (continuity, a priori knowledge).

It is necessary to maintain a good diversity in the population for the algorithm. Algorithms that work with several

subpopulations work better (because there is more diversity). The actual coding is suitable for parametric optimization problems.

When a controller is implemented with genetic algorithms, the general structure of a control system can not be lost because the system error and manipulated variable can be used as objective functions to find the best solution.

The genetic algorithm works because there is competition for resources and inherit the best genetic configuration in each generation. Certain characteristics of an organism make it more successful than other organisms of the same species, where success is defined as reproductive success, the better adapted leaves more children than the others. Heritable traits that favor success tend to be more represented in the population.

ACKNOWLEDGEMENT
This paper is funded by Central University of Bogota Colombia.

REFERENCES

[1] Neuro-Fuzzy and Soft Cornputing. Jang Sun y Mizutani. Prentice Hall. 1997. ISBN 0-13-261066-3.
[2] Redes Neuronales y Sistemas Difusos. Martin del Brio, Bonifacio y Sanz Molina, Alfredo. Alfaomega. 2001. ISBN 970-I5-033-9
[3] Inteligencia Artificial y minirobots. Delgado, Alfredo. Ecoe Ediciones. 1998. ISBN 958-648-155-7.
[4] An Introduction to Fuzzy Sets. Analys1s and Design. Pedrycz Witold y Gomide, Fernando. MIT.1998. ISBN 0-262-16171-0.