Genetic algorithms in PD Control Systems, a multivariable and multiobjective approach

Edgar Chavolla, Erik Cuevas, Daniel Zaldivar, Marco Perez and Alberto De La Mora
chavolla@gmail.com
Universidad de Guadalajara, CUCEI, Av. Revolución 1500,
44430 Guadalajara, Jalisco, México.

(Paper received on February 29, 2008, accepted on April 15, 2008)

Abstract.

The design of a PID controller is a multiobjective problem. The designer has to adjust the controller parameters such that the feedback interconnection of the plant and the controller satisfies a set of specifications. The specifications are usually competitive and any acceptable solution requires a trade-off among them. In this work an approach for adjusting the parameters of a PD controller based on multiobjective optimization and genetic algorithms is presented. This approach was proven successfully to find the parameters of a PD controller on a level plant

1 Introduction

In recent years the development of controllers on the base of genetic algorithms has been paid much attention. The PID is the most accepted controller in the industry. In fact, most of them are PD controllers because the integral action has been switched off. Although the number of parameters to adjust in a PID controller is very small and a great deal of tuning rules can be found in the literature [1]. In a recent study, it has been experimentally checked that more than 30% of the installed controllers are operating in manual mode and 65% of the loops operating in automatic mode are poorly tuned [2]. This justifies the search for new approaches to adjust industrial controllers.

During the past decades great attention has been paid to optimization methods for controller design. The control design problem is a multiobjective problem. An effective design method should allow one to deal with several objectives that could possibly be expressed using various types of norms.

The fixed structure of the PID controllers creates serious problems for applying the modern optimal design methods that deal with unstructured controllers. Moreover, the resultant optimization problem is not convex and local optimization methods can be stuck in a local minimum. This has motivated the use of genetic algorithms GA's for adjusting PID controllers [3-5].

Special Issue in Electronics and Biomedical Informatics,
Computer Science and Informatics
Research in Computing Science 35, 2008, pp. 41-50

The genetic algorithm is a method for solving both constrained and unconstrained optimization problems that is based on natural selection. The genetic algorithm repeatedly modifies a population of individual solutions. At each step, the genetic algorithm selects individuals at random from the current population to be parents and uses them to produce the children for the next generation. Over successive generations, the population "evolves" toward an optimal solution. Genetic algorithm can be applied to solve a variety of optimization problems that are not well suited for standard optimization algorithms, including problems in which the objective function is discontinuous, multiple, nondifferentiable, stochastic, or highly nonlinear.

In this work an approach for adjusting the parameters of a PD controller based on multiobjective optimization and genetic algorithms is presented. This approach was proven successfully to find the parameters of a PD controller on a level plant

The document is organized as follows: in section 2 the plant is described, in section 3 the optimization approach is explained, in 4 the results are presented and finally in section 5 the conclusions are established.

2. System description

The plant in this work is a water level system. This system is controlled by a PD Fuzzy system [6]. The PD Controller has three gain parameters to adjust the behavior of the control. Every parameter has an effect over the system behavior. In the water level System there are three objectives identified to be considered in the output level: Maximum Peak level (MP), Time to reach the desired level (TC), Time for Stabilization (TSS). These objectives are shown in the figure 1.

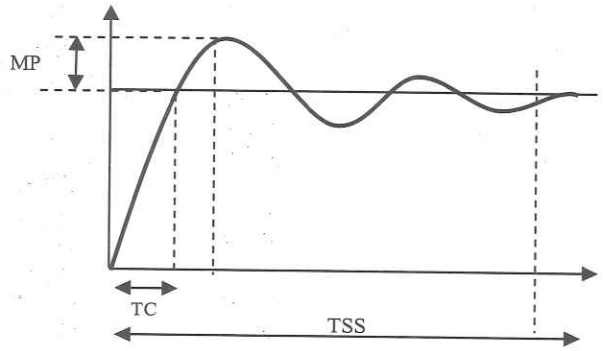

Fig. 1. System response objectives.

Our whole system has the following configuration: The reference level block, the error signal and error rate block, the PD Fuzzy Block, the output signal block and the plant. Figure 2 shows the system configuration.

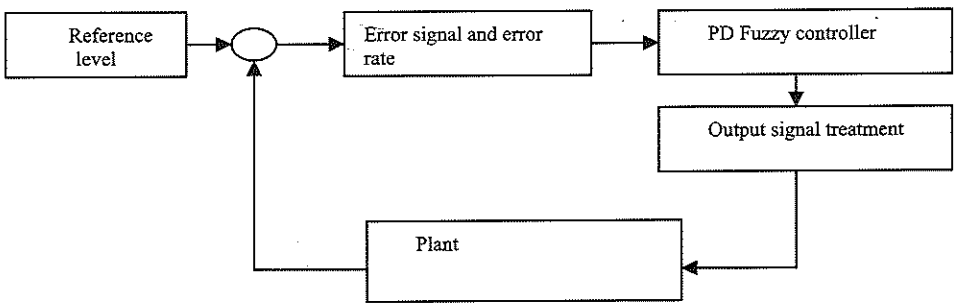

Fig. 2. System configuration

There are inside the "error signal, error rate" and the "output signal" blocks three gains that affect lineally the system. The gains are for the input: Error Gain (Ge) and Rate Gain (Gr), and Output Gain (Gu) [7] for the output.

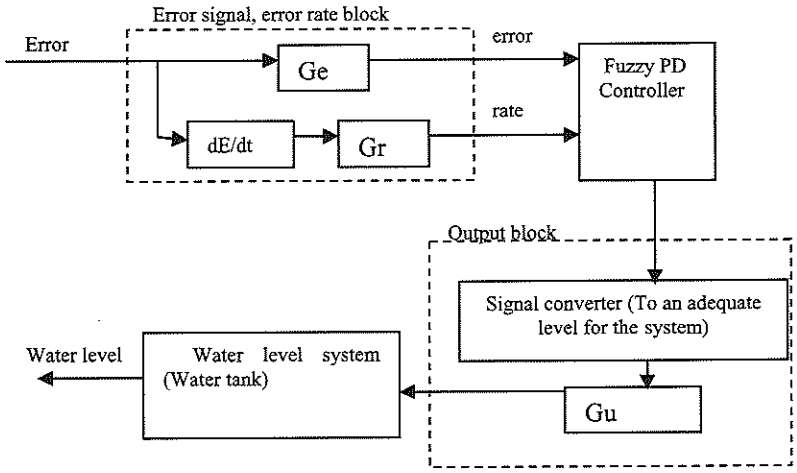

Fig. 3. A closer look into the target system. Here the gains can be located (Ge, Gr and Gu).

Ge, Gr and Gu will be the input variables for the GA, while TC, MP and TSS will be the objectives to achieve.

3 GA procedure

Multi Objective Problems (MOP) is a field where in the latest years has been many advances and new algorithms have been developed and proposed. The general field where all these algorithms have been classified is named Multi- Objective Evolutionary Algorithm (MOEA) [8].

In the process to solve the multi-variable algorithm and the MOP, some techniques that allow improving the effectiveness of the algorithm will be used. These techniques converge in a faster way to a convenient solution.

The problem can be divided in three main sections: A general GA implementation, the Multi-Objective Algorithm (MOA), and the Multi-Variable Algorithm (MVA)

The general implementation procedure contains the following operations:

1. Set parameters
2. Create randomly a population
3. loop until condition
4. mix population
5. crossover
6. mutation
7. evaluate fitness
8. natural selection
9. end loop
10. Show results

MOA will act in the fitness evaluation process (step 7), and MVA will act in the crossover process (step 5). The rest of the steps can be created as any normal GA.

3.1 General GA implementation

This process is almost the same as any other GA; however is needed to add some specific parameters that help to the MOA and MVA procedures. The crossover step will not take place in the general GA implementation, it will be done in the MVA procedure, and the fitness evaluation will be completed by the MOA procedure.

For the MVA, three random seeds will be created (one for every variable), so it can have independent values to create or mutate each gene. Also for the MOA a *dominance factor* is needed; this will tell the crossover algorithm (the MVA) how to mix the genes. In MOA the parameters are set to the desired values for the objectives, and the weight of them.

3.2 MVA implementation (Crossover)

In this part the inheritance theory from Gregor Johann Mendel is used. The Mendel's Theory states a set of primary tenets relating to the transmission of hereditary characteristics from parent organisms to their children

This theory basically tells the way the parent's genes are affecting genes in the offspring. There are two types of genes in the parents, the dominant genes and the recessive genes. The dominant genes will affect mostly in the offspring, while the recessive gene will have little effect. Enumerating some cases from the gene combination based on the dominant factor (Table 1).

Parent 1 gene	Parent 2 gene	Offspring gene
+X	-Y	X(Y) Mainly affect by X
-X	+Y	Y(X) Mainly affect by Y
+X	+Y	XY Equally affected by X and Y
-X	-Y	XY Equally affected by X and Y
++X	--Y	X Only affected by X
--X	++Y	Y Only affected by Y

Table 1. The possible combinations from the genes considering the dominance factor. + means a dominant gene, ++ means a superior gene, while – means a recessive gene, and -- a gene that is likely to disappear.

Table 1 shows that there are 5 different kinds of breed from the parents. This table applies for every gene in the chromosome. It is needed to consider all the possible combination, because it cannot be decided a priori what would be the dominant gene. This is due to the Multi-objective problem; cannot be assured if giving more weight to a gene will improve or will degrade one or some of the objectives.

Based on table 1, the total number of the new breed for each pair of individuals can calculate as:

$$offspring\_number = number\_of\_combinations^{number\_of\_genes} \quad (1)$$

In the system there are 3 genes and using 5 combinations, this will give using equation (1), 125 new offsprings in every generation for every parent pair. This is a big breed, but eliminating some the possible combinations can reduce it.

For the combination of the genes, it will be used Eq. (2), so the general definition of the crossover is an arithmetic operation. The values for *gain*_1 and *gain*_2 correspond to the dominance factor defined for the gene.

$$new\_gene = \left(gene\_1 * gain\_1 + gene\_2 * gain\_2\right)/2 \quad (2)$$

The dominant factor will be multiplied by *gene_* 1 or *gene_*2, depending on the combination and the way the algorithm needs to converge (table 2).

Parent gene 1	Parent gene 2	Formula for the offspring gene
+X	-Y	(X * (Dominant_factor) + Y) /2
-X	+Y	(Y * (Dominant_factor) + X) /2
+X	+Y	(X + Y) /2
-X	-Y	(X + Y) /2
++X	--Y	X
--X	++Y	Y

Table 2. This table shows the formulas used depending on the gene combination.

Considering the previous tables and equations, the MVA can be described as:

1. Take pairs of individuals (parents) from the population
2. Create the number of children according to the combination formula (1)
3. Assign values to the genes depending on the combinations done base on Table 1 using the formulas given by Table 2

3.3 MOA implementation (Evaluate fitness)

This process operates with each individual that has not been yet evaluated. In the water level plant, the objective functions are measures from the output response. As the controller affects the system, it is relatively easy to define equations that evaluate the objectives of the system dynamics.

The solution is to accomplish an implementation of the whole system, and set the gain values using the data contained in the genes of each individual. So for each individual it will be run a simulation of the whole process in order to obtain the objective values.

After evaluating the objective values with the data from the individual, the output values are compared against the desired values, and the error rate is obtained. The equation (3) can be used if the desired value is not zero. If the desired value is zero, the error rate can be approximated by equation (3a). The correction value in equation (3a) should be set by testing; the value should be set between 0 and 1. This correction value prevents big values in the error rate.

$$error\_rate = \left(\left|desired\_value - obtained\_value\right|\right)/desired\_value \tag{3}$$

$$error\_rate = \left|obtained\_value\right| * correction\_value \tag{3a}$$

Equations (3) and (3a) produce three error values (one for each objective). Now the error values are combined in a fitness value, so this fitness value ranks the individual behavior obtained.

The next step involves the weight values set for every objective. These weigh values define how strong is the objectives in the algorithm. Thus, a stronger objective will affect more the fitness values than one with a low weight.

In MOP the fitness value is calculated from the three objectives using (4), this expression calculates the weight mean depending on the weights values and error rates of every objective (lower fitness values are considered to be better).

$$fitness = \frac{(error\_1 * weight\_1 + error\_2 * weight\_2 + error\_3 * weight\_3)}{(weight\_1 + weight\_2 + weight\_3)} \quad (4)$$

The process is summarized as:

1. Evaluate the system for every individual in the population
2. Get the objective values for each individual
3. Calculate the error rate for every objective
4. Calculate the fitness values

4 Results

The implementation of the GA was made in Matlab and Java. Java was chosen due to the nature of the problem that can be mapped directly to OOP; and also because it is supported by Matlab.

In the water level plant, MP parameter is defined as the absolute difference between the maximum real value and the desired value. This is a little different from the original definition. This change makes possible support the case, when the output is under the desired level.

Considering the importance of the parameters, the weights must be selected. MP is the most important parameter as well as the TC parameter, while the TSS parameter is not considered as critical in the application. In order to set TC and TSS to the desired values, the minimum filling time for the water plant should be considered. This is around to 162 seconds, so TC an TSS can not be less then this value.

Parameter	Value
Weight for MP	1
Weight for TC	1
Weight for TSS	0.1

Population size	10
Dominance factor	0.2
Mutation factor	50%
Desired level	60
Seed for Ge	140
Seed for Gr	2
Seed for Gu	3
Desired MP	.01
Desired TC	170
Desired TSS	180

Table 3. The parameter used in the GA implementation.

After running the GA 30 generations, acceptable results were found (Table 4). TSS is exactly the required value and a better Mp value was found (closer to the desired level), only TC was higher than expected. This can be considered a good solution to the plant (only 0.0886 in the fitness value).

Parameter	Desired values	Obtained values
MP (weight 1)	.01	.0085
TC (weight 1)	170	174
TSS (weight .1)	180	180

Table 4. The result from the tuned up parameters after 30 generations

An optimal result was found after only 18 generations, which can be considered a fast way to get the gain values for the PD controller (Fig.4). The gain values were Gu: 1.674270492132032, Gr: 83.5564484964882, and Ge: 75.69871513072385.

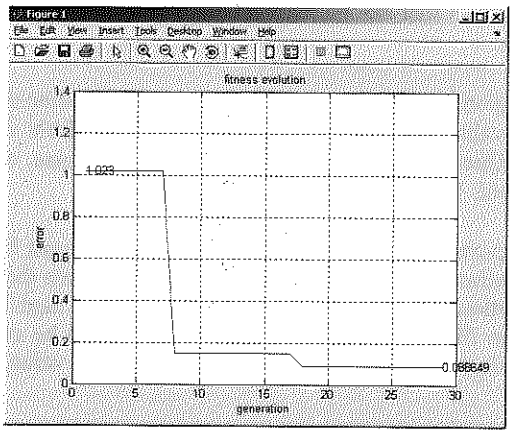

Fig.4. The fitness function shows how an optimal result was found after 30 generations.

5 Conclusions

In this work an approach for adjusting the parameters of a PD controller based on multiobjective optimization and genetic algorithms was presented. This approach was proven successfully to find the parameters of a PD controller on a level plant

PD and PID controller systems can be greatly improved using the techniques described in this paper. The controller can be set to operate in a desired operation range with a minimum of error.

Moreover, the technique used in the controller finds an optimal set of gain values faster than manual procedure or other techniques. This is true most of the time, due to the randomness in the nature of the technique itself. Some times the GA would find a local solution, so the algorithm should be stop and started again. Doing this restating process will allow the GA to find a better place to converge to a good solution.

To find a good solution the dominance factor was set to a level that let the GA to change in fast way without losing precision (a high value make the GA converge faster, and low values make the GA get more precise values). It was found that this value can be between 0.1 and 1 to find optimal precision-speed performance.

The seeds for Gr, Ge and Gu were chosen using previous experiences in the system, so the random values were generated close to some values that give a stable behavior. These seeds guaranty that the GA will start around a value that can be closer the desired. The mutation factor was chosen to 50% to let the GA have plenty of new points to search without being a random search with not a tendency.

The problem using GA-MVA-MOA, is the amount of resources needed to make it run. The memory and the computer speed affects greatly in the GA performance. As a consequence a really careful setting in the initial parameters in the GA should be done in order to get good results and have a good performance in the GA.

The population was set to 10, this was due to limitations of resource in the computer, also 10 is wide enough to let the GA have a good number of combinations without overwhelming the algorithm.

The incorporation of MVA and MOA in GA allows to find optimal gain values faster than other methods and to reach the global minimum.

References

[1] O'Dwyer, A.: PI and PID Controller Tuning Rules for Time Delay Processes: a Summary. Tech. Rep. AOD-00-01 Ver. 1, Dublin Institute of Technology, Ireland (2000)

[2] Ender, D. B.: Process control performance: Not as good as you think. Control. Eng. 40 -10, 180–190 (1993)

[3] Jones, A. and De Moura Oliveira, P.: Genetic autotuning of PID controller. In Proceedings of the First International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications, Sheffield, UK, IEE, (1995) pp. 141–530.

[4] Salami, M. and Cain, G., An adaptive PID controller based on genetic algorithm processor. In Proceedings of the First International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications, Sheffield, UK, IEE, (1995), pp. 88–94.

[5] Chen, B. and Cheng, Y., A structure-specified $H^{\cdot}$ optimal control design for practical applications: A genetic approach. IEEE Trans. Control Syst. Technol, (1998) pp. 707–718.

[6] Sanchez E., Nuno L.A., Hsu Y.C., and Guanrong C, Real Time Fuzzy Swing-up Control for an Underactuated Robot, *JCIS '98 Proceedings, Vol 1* , N.C., USA, (1998)

[7] Cuevas E. V., Zaldívar D., and R. Rojas.: Incremental fuzzy control for a biped robot balance. IASTED International Conference on ROBOTICS AND APPLICATIONS ~RA 2005, Cambridge, USA (2005) pp.7-12.

[8] Carlos A. Coello Coello, Gary B. Lamont and David A. Van Veldhuizen, Evolutionary Algorithms for Solving Multi-Objective Problems (Second edition), Springer, (2007)