An Adaptive PID Controller Based on Genetic Algorithm Processor

Mehrdad Salami and Greg Cain

Victoria University of Technology, Australia

Email: salami@cabsav.vut.edu.au , gregc@cabsav.vut.edu.au

Abstract - **This paper presents a method of designing Proportional-Integral-Differential (PID) Controllers for single-input, single-output (SISO) systems. We used a Genetic Algorithm implemented in hardware for setting the K values of the PID Controller and minimizing the integral error in the system. The system have been applied to a PID Controller system with variable plant transfer function. The method has been successfully tested and some results are presented.**

I. INTRODUCTION

The conventional PID regulator is the most widely used control strategy in the process industry. This is because of the remarkable effectiveness, simplicity of implementation and broad applicability.

Often in practice, tuning is carried out by an experienced operators using a 'trial and error' procedure and some practical rules. This is often a time consuming and difficult activity, for example when the dynamic process is slow, partly nonlinear, contains significant dead-time or when there are random disturbances acting on the plant. Once tuned, the control performance may later deteriorate because of nonlinear or time-varying characteristics of the process under control. Although PID regulators are common and well-known, they are often poorly tuned. Evidence of this can be found in almost any industrial process.

Modern adaptive control algorithms can be a good solution to such problems. They are able to self-tune the regulator and to adapt it to changes in the process, provided certain conditions are fulfilled [1,2,3].

When these controllers are introduced in industry, some resistance and difficulties can arise, mostly related to a lack of knowledge about their internal mechanisms by the operating personnel. An attractive alternative is to try to combine the well-known PID Controller with algorithms which are able to provide, on-line, a set of optimal PID parameters using input/output data from the system.

The genetic algorithm is very effective at finding optimal solutions to a variety of problems. This innovative technique performs especially well when solving complex problems because it doesn't impose many limitations of traditional techniques. Due to its evolutionary nature, a GA will search for solutions without regard to the specific inner working of the problem. This ability lets the same general purpose GA routine perform well on large, complex scheduling problems. There is now considerable evidence that genetic algorithms are useful for global function optimization and NP-Hard problems [4,5].

For a genetic algorithm to improve a solution, it is necessary to reject the poor solutions and only allow reproduction from the best ones. This, in turn, is again an analogy to the survival of the fittest law, which only allows an organism that adapts best to the natural environment to survive. In this case, the role of the environment is played by a so called evaluating function, measuring the degree of fitness of a candidate to problem requirements. This is equivalent to a function which tests whether a given state is a terminal one. Genetic algorithms can be applied to optimization problems or in artificial intelligence.

In this paper firstly the PID Controller will be explained with its objective function for optimization. Next we will describe the genetic algorithm principles and how a Genetic Algorithm Processor (GAP) can optimize a control module. Then a Multiple Genetic Algorithm Processor configuration will be explained. Finally we will apply our model to see how the GAP can find the best

Genetic Algorithms in Engineering Systems: Innovations and Applications
12-14 September 1995, Conference Publication No. 414, © IEE, 1995

combination of K values for the PID Controller.

II. A PID CONTROLLER

Designing PID Controllers, even for low order plants such as a robot arm, can be a difficult problem. Consider the system illustrated in Figure 1 where the PID Controller obeys the following control law:

$$M(S) = \left(K_i / S + K_p + K_d * S\right) E(S)$$

It can be expressed in the Z domain as:

$$M(Z) = \left(K_i / \left(1 - Z^{-1}\right) + K_p + K_d * \left(1 - Z^{-1}\right)\right) E(Z)$$

In the equations Ki is the integrator gain coefficient, Kp is the proportional gain control and Kd is the differentiator gain control. The goal of PID Controller design is to determine a set of gains, (Ki, Kp, Kd), of the control law such that the set of roots of the characteristic equation chosen by the designer are obtained. The three gain parameters (Ki, Kp, Kd) of the PID control law interact with the plant parameters ***P(S)*** in a complex fashion when the designer attempts to obtain the specified roots of the characteristic equation. These roots are chosen in order to obtain the desired transient response of the closed loop, while taking the resultant zeros into account. PID Controllers increase the order of the characteristic equation by one. The controller introduces a new pole at the origin of the s-plane, and they shift the original compensated roots of the closed loop system to new positions on the s-plane. In addition to these effects, PID Controllers introduce a pair of zeros, usually a complex conjugate pair, which will normally have a significant effect on the transient behaviour of the compensated system.

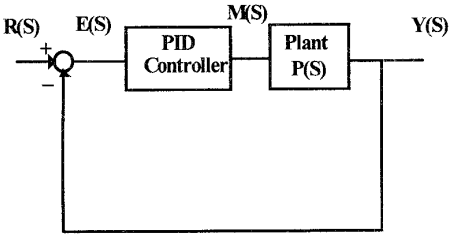

Figure 1 . A typical PID Control system

The efficiency of the system can be measured by calculating the integral of the time multiplied by the error for the unit step response during [0,T]:

$$error = \int_{t=0}^{T} t\,|e(t)|\,dt$$

The problem confronting the designer, therefore, is to calculate the three gains of the PID Controller while ensuring that transient response (minimum error, overshoot, rising time, settling time and steady-state error) specifications are met.

III. GENETIC ALGORITHM PROCESSOR

A genetic algorithm (GA) is a robust optimization technique based on natural selection. The basic goal of GAs is to optimize functions called fitness functions. GA-based approaches differ from conventional problem-solving methods in several ways [6]. First, GAs work with a coding of the parameter set rather than the parameters themselves. Second, GAs search from a population of points rather than a single point. Third, GAs use payoff (objective function) information, not other auxiliary knowledge. Finally, GAs use probabilistic transition rules, not deterministic rules. These properties make GAs robust, powerful, and data-independent.

A simple GA starts with a population of solutions encoded in one of many ways. Binary encodings are quite common and are used in this report. The GA determines each string's strength based on an objective function and performs one or more of three genetic operators on certain strings in the population.

1. Reproduction (also called selection) is simply retaining a fit string in the following generation.
2. Crossover involves swapping partial strings of random length between two parent strings.
3. Mutation involves flipping a random bit in a string.

These three operations primarily involve random number generation, copying, and partial string exchange. Thus GAs are simple to implement.

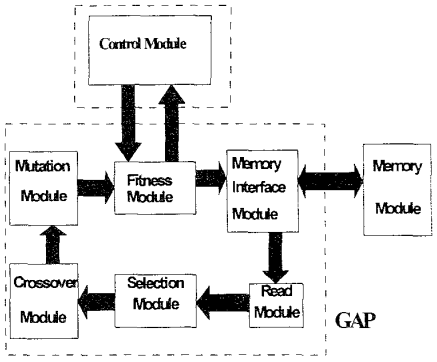

Figure 2 . Internal architecture of the GAP

A genetic algorithm processor can be constructed to directly execute the operation of a genetic algorithm [7,8,9]. It is shown in the Figure 2. Such a processor can be used in situations where high throughput is required and where the logic of the genetic algorithm is expressible in simple units which can be synthesised in hardware. This is generally the case as genetic algorithms are inherently simple and contain only a few logic operations.

IV. Multiple GAPs

A critical parameter when using a GAP in solving a complex problem is the bit length of operation. Normally complex problems require long bit length operations inside the GAP. On the other hand because the GAP is implemented in hardware long bit lengths lead to high cost and low processing speed. From the GAs point of view higher bit length means more searching in solution space and more time to solve a problem. One solution to all these difficulties is dividing the bit length across multiple GAPs. In this model each GAP is responsible to optimize one part of the total bit length [8].

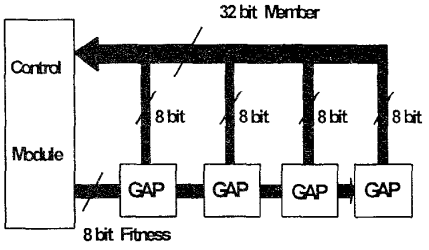

Figure 3 . The architecture of a multiple GAP

Figure 3 shows the configuration of multiple GAP. Each GAP delivers its bit slice to the control module. In the control module the bit slices combine with the best member of other GAPs to form a complete genetic member.

V. Simulation Results

Genetic Algorithms Processor simulations have been conducted for the PID Controller system in Figure 4. The reference signal (R(t)) is a step signal and it is shown in Figure 5. The transfer function for plant is equal to [10]:

$$P(S) = \frac{a}{S(S-1)(S+5)} \qquad \text{Eq. 1}$$

In this transfer function value 'a' is made to drift from 1 in the beginning to 30 during simulations. The PID Controller transfer function is defined as:

$$U(S) = \left(K_i / S + K_p + K_d * S\right) E(S) \quad \text{Eq. 2}$$

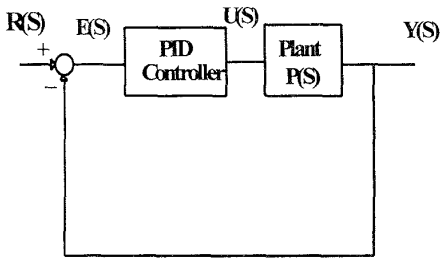

Figure 4 . A typical PID Controller system

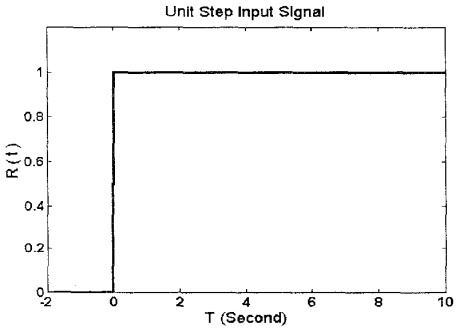

Figure 5 . The reference signal

To handle this problem we used Multiple GAPs. The arrangement of Multiple GAPs is depicted in Figure 6. Each GAP optimizes one

K value from the PID Controller. The following parameters are selected for all GAPs:

3000 = Number of generations.
8_bit = Representation of the fitness value.
64 = Population size.
0.9% = Probability of crossover.
0.02% = Probability of mutation.

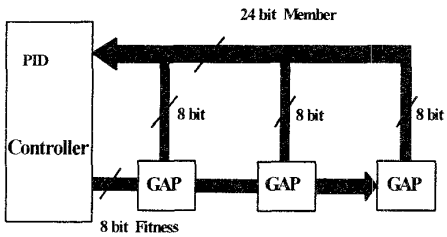

Figure 6 . The architecture of the Multiple GAP used for PID Controller system.

To illustrate drifting in the 'a' value of the plant transfer function we incremented the value by one after each 100 generations according to Figure 7.

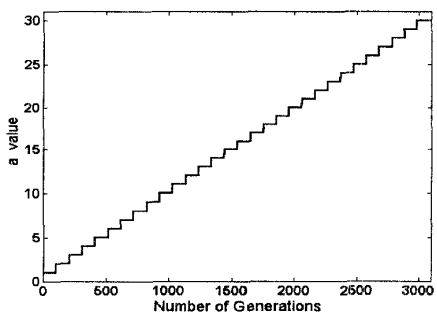

Figure 7 . The 'a' value changes with the Number of Generations

For calculating the fitness value we converted all transfer functions from the S domain to the Z domain and then to the discrete values (K domain). For simulations we selected 2000 points between 0 an 10 second (T = 10/2000 = 0.005). Then the integral of the time multiplied by the absolute error values are calculated as the fitness value.

$$Fitness = \sum_{k=1}^{2000} \left((kT) * |e(k)| \right) \qquad \text{Eq. 3}$$

Figure 8 shows the result of simulation averaged over ten individual runs. In the figure the best of the three K values and the fitness values are shown after each generation. Figure 9 shows the average best response signal (Y(t)) for a=1 and a=30. The characteristic of the response signal is shown in Table 1.

Parameters	a=1	a=30
Kp	206	23
Ki	131	14
Kd	38	15
Steady State Error	0.000	0.000
Overshoot	% 0.7676	% 1.122
Rising Time(S)	1.150	1.495
Settling Time(S)	1.575	1.600
Integral of Absolute Error	90.36	92.26

Table 1 - Characteristic of the best member for a=1 and a=30

In the Figure 8 Kp changes from about 200 for a=1 to 30 for a=30. In the same time Kd changes from 130 to 20 and Ki changes from 35 to 5. This shows when a is changing in the system the Kp in the PID Controller has the most effect on adjusting the system. The fitness value graph shows that the GAP always keeps the response signal whitin a minimum error during changes in the a value. Figure 9 shows the response signal always remains with the high quality.

VI. CONCLUSION

In this paper we have described a model of a digital processor which embodies a universal genetic algorithm. We have demonstrated the performance of this model on a PID Controller system. A new Multiple GAPs has been introduced and we showed that the Multiple GAPs can be applied to a PID Controller in a dynamic plant system. The graphs demonstrate the high performance of Multiple GAPs in finding good K values for the PID Controller.

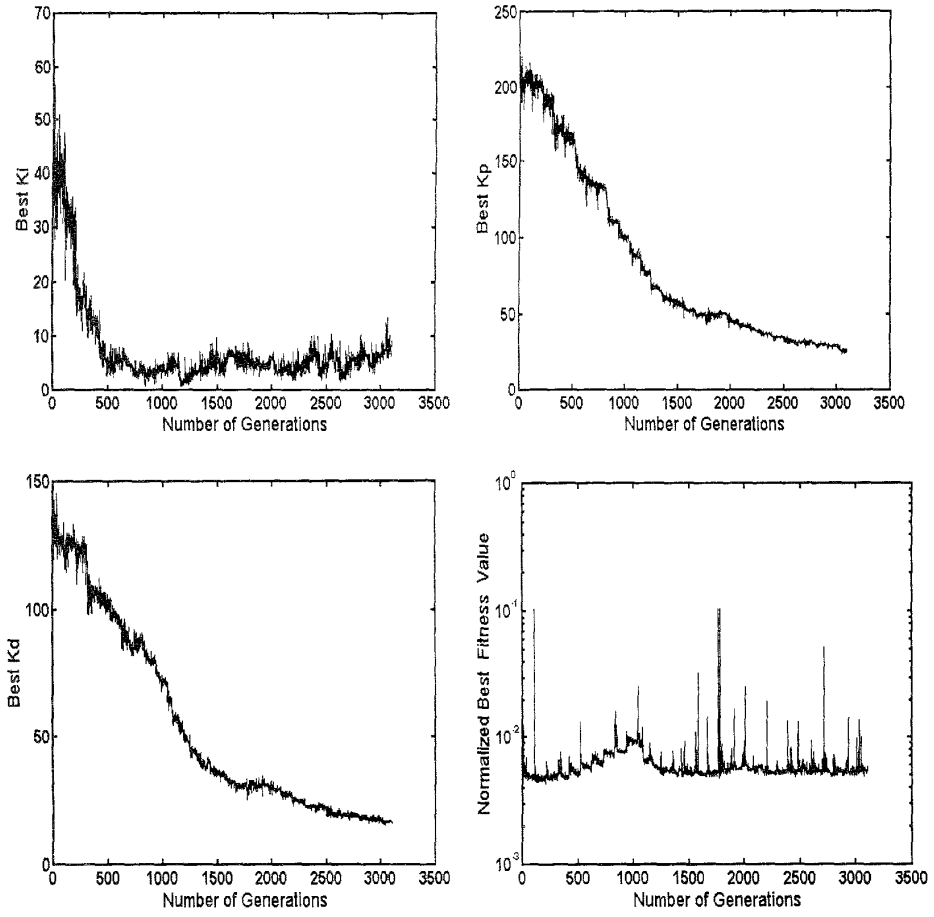

Figure 8 . The results of the PID Controller simulation. The best Kp, Kd, Ki and the fitness values versus Number of Generations

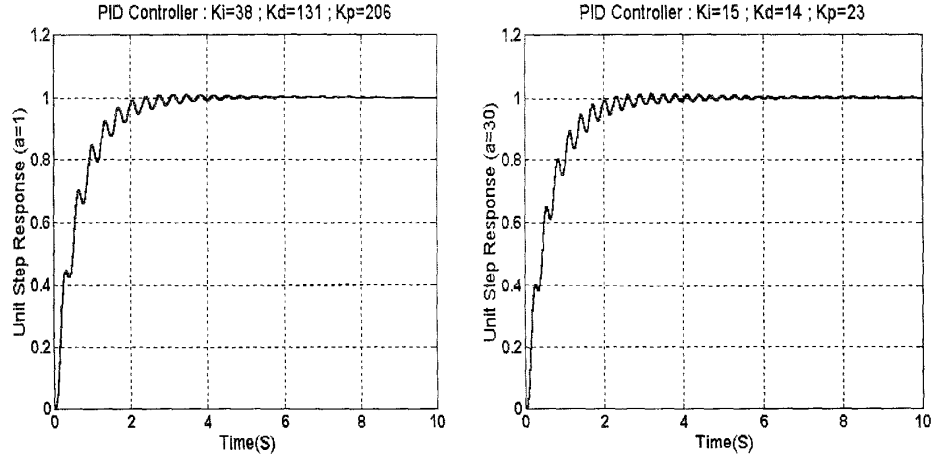

Figure 9 . The Unit Step Response of the PID Controller system for a=1 and a=30

REFERENCES

[1] Ogata, K. , "Modern Control Engineering " , Englewood Cliffs. NJ, Prentice-Hall 1970.

[2] Dorf, R.C., "Modern Control Systems", Addison-Wesely Publishing, Reading, MA, 6th Eddition, 1991.

[3] Paraskevopoulos, P. N., "On the Design of PID Controller for Linear Multivariable Systems", IEEE Transactions on Industrial Electronics and Control Instrumentation, Vol. IECI-27, No. 1, Feb. 1988.

[4] Goldberg D. E., "Genetic Algorithms in Search, Optimization and Machine Learning", Addison-Wesley Co, 1989.

[5] Holland J.H. , "Genetic Algorithms", Scientific American, July 1992.

[6] Davis, L., "Handbook of Genetic Algorithms", International Thomson Publishing, New York 1991.

[7] Salami, M., Cain, G., "The Quest for a New Computing Architecture Based on Genetic Algorithms", Proceedings of the Electrical Engineering Congress (EEC94) , Sydney, Australia, November 1994.

[8] Salami, M., Cain, G., "An Adaptive Control System Based on Genetic Algorithms", Proceedings of the FLAIRS-95 International Workshop on Intelligence Adaptive Systems (IAS-95), Melbourne Beach, Florida, April 1995.

[9] Salami, M., Cain, G., "Adaptive Hardware Optimization Based on Genetic Algorithms", Proceedings of The Eighth International Conference on Industrial Application of Artificial Intelligence & Expert Systems (IEA95AIE) , Melbourne, Australia, June 1995.

[10] Hwang, W.R., Thompson, W.E., "An Intelligent Controller Design Based on Genetic Algorithms" , Proceedings of the 32nd Conference on Decision and Control, San Antonio, Texas, December 1993.