A novel Implementation Technique for Genetic Algorithm based Auto-Tuning PID Controller

A. Concha[§], E. K. Varadharaj[†], N. M. Hernandez-Rivera[‡], and S. K. Gadi[¶*]
[§]FIME, Universidad de Colima, Coquimatlán, Colima, México. Email: aconcha@ucol.mx
[†]Indian Institute of Information Technology Design and Manufacturing, Kurnool, India. Email: Eswaramoorthykv@iiitk.ac.in
[‡]Facultad de sistemas, Universidad Autónoma de Coahuila, Saltillo, Coahuila, México. Email: noehernandez@uadec.edu.mx
[¶]FIME, Universidad Autónoma de Coahuila, Torreón, Coahuila, México. Email: research@skgadi.com
[*]Address all correspondence to this author.

***Abstract*—The auto-tuning proportional-integral-derivative (PID) controllers based on genetic algorithms have many limitations in their implementation. For example, some chromosomes of the tuned genetic algorithms may cause closed-loop system instability, and some chromosomes tuned for an operating condition may cause instability for other operating conditions. This article provides the possible solutions for these problems. An implementation technique of a genetic algorithm is proposed based on these solutions. Numerical simulations are given to evaluate the proposed method applied to a system with variable parameters. Moreover, this technique is also used for controlling a brushless servo motor.**

***Index Terms*—Genetic algorithm, PID tuning**

I. Introduction

Theoretically, a proportional-integral-derivative (PID) controller guarantees stability for lower order linear systems as servomechanisms [1]. Moreover, this controller is used in many industries to control nonlinear and delayed systems [2], [3]. A nonlinear system behaves like a linear system under certain operating conditions [4]. Hence stability is achieved by a PID controller for a particular operating condition. The PID parameters must be updated when the operating conditions change for a stable operation. In the case of a delayed system, apart from the PID parameters, the transportation delay also plays a significant role in its stability. In some cases, introducing an additional delay into the system stabilizes it [5].

Tuning PI and PID controllers has been a topic of research during the last three decades [6]. It has been carried out using different techniques such as fuzzy rules [7], SIMC PID tuning rules [8], the Lyapunov approach [9], fuzzy neural networks [10], and genetic algorithms [11], [12]. Among of these techniques, genetic algorithms may tune a PID controller that produce an unstable closed loop system. This paper proposes the solution of this problem, as well as a genetic algorithm providing a stable closed loop system. The performance of the proposed algorithm is evaluated through numerical simulations and experiments. The paper is structured as follows. The PID controller is presented in section III. Section IV describes the Conventional Genetic Algorithm based PID tuning. Its limitations and proposed solutions of them are mentioned in section V. The proposed genetic algorithm based on these solutions is presented in section VI. Simulation and experimental results are shown in sections VII and VIII, respectively. The paper ends with some concluding remarks.

II. Nomenclature

C	Population	$\mathbb{R}^{n\times 4}$
C_F^-	The previous generation's fittest chromosome	$\mathbb{R}^4$
C_i	i^{th} chromosome of the population	$\mathbb{R}^4$
e	Error between reference and measured plant output	
E_A	Maximum allowed E_i to test the termination criterion for the tuning process	
E_i	Root mean square error, which is the value for the i^{th} chromosome's cost or objective function	
E_M	Upper limit for the E_i, beyond which the parameter's values are changed from the i^{th} chromosome to C_F^-	
K_D	Derivative constant of PID controller	
K_I	Integral constant of PID controller	
K_P	Proportional constant of PID controller	
M_E	Moving root mean square of the error e	
M_E^+	Maximum allowed M_E, beyond which tuning process restarts	
N	Constant associated with the filter implementing the derivative term in a PID controller	
n	Population size	
r	Reference signal	
s	Laplace transform's complex variable	
T	Time interval for which the i^{th} chromosome is applied	
t	Time	
T_i^+	End time of the implementation of the i^{th} chromosome	
T_i^-	Start time of the implementation of the i^{th} chromosome	
T_s	Empirically estimated settling time	
T_w	Window size for calculating moving average	
u	Control signal	
y	Output of the plant	

III. PID controller

Fig. 1 shows a typical closed loop system with negative feedback. The plant and controller blocks represent its dynamics. $r(t)$, $u(t)$, $y(t)$ and $e(t) = r(t) - y(t)$ are reference,

978-1-5386-0814-2/17/$31.00

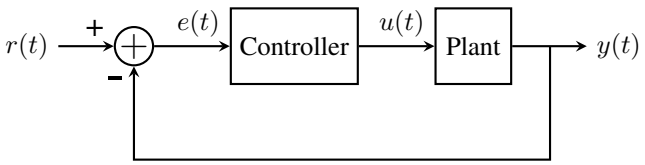

Fig. 1. Closed loop system with negative feedback

control, output and error signals respectively. The objective of a controller is to achieve

$$\lim_{t\to\infty} e(t) = 0.$$

A PID controller is defined as

$$u(t) \quad = \quad K_P e(t) + K_I \int e(t)\,\mathrm{d}t + K_D \frac{\mathrm{d}}{\mathrm{d}t}(e(t)), \quad (1)$$

where K_P, K_I, and K_D are nonnegative constants. These parameters are tuned to achieve stability with required performance. The required performance can be ensuring settling time, providing damping or improving raise time, etc.

Usually, sensors are used to measure the output of the plant $y(t)$ which are prone to noise. The noise power is amplified when differentiation is performed [13]. Therefore instead of using the PID controller 1, the third term associated with the derivative of $e(t)$ is replaced with a high pass filter of $e(t)$. Modified version of PID controller in frequency domain is

$$u(s) \quad = \quad K_P e(s) + \frac{K_I e(s)}{s} + \frac{K_D N e(s) s}{s+N}, \quad (2)$$

where s is the Laplace transform's complex variable and $N >> 1$ is a constant.

Tuning a PID controller is a task of selecting optimal values for the four parameters K_P, K_I, K_D, and N for which the plant is stable. In practice, N is kept constant. However, in this article N is considered as one of the tunable parameters.

The next section presents a typical method for tuning these parameters with the help of a genetic algorithm.

IV. Conventional Genetic Algorithm based PID tuning

Fig. 2 shows a standard genetic algorithm based PID tuning algorithm [11], [12]. In this context, the word population is used to represent a fixed set of individuals or chromosomes; and an individual or a chromosome represents an array of values for the parameters to be tuned. In this case the parameters are K_P, K_I, K_D and N. Each chromosome is a candidate for the optimal solution. The genetic algorithm mimics the evolution process to generate optimal values for the parameters. It starts with a generation of a random population; it undergoes reproduction and mutation process to produce a next generation population with values much closer to the optimal values. This algorithm includes the following operations.

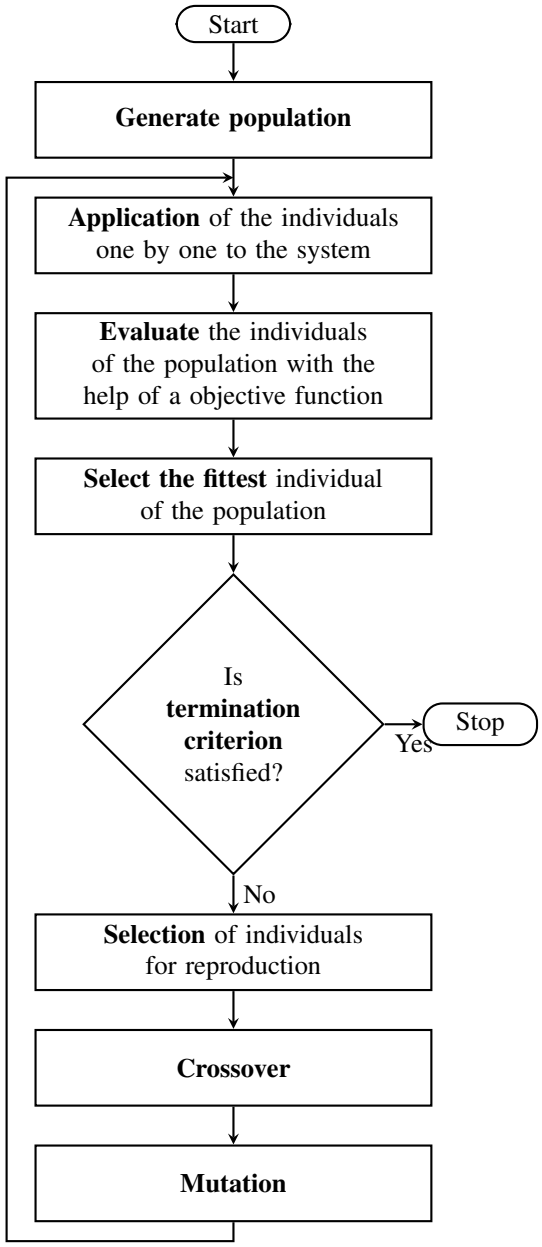

Fig. 2. Conventional genetic algorithm based PID tuning algorithm

a) Generate population: A predetermined number of random chromosomes, $C \in \mathbb{R}^{n\times 4}$, are created to group them as a first generation population. The constant n represents the population size. The random numbers are scaled to match the upper and lower limits of the parameters. In this case, the random numbers must be greater than zero.

b) Application: The values for the tunable parameters of the closed-loop system, K_P, K_I, K_D and N are replaced by each of the present generation's chromosomes, $C_i\mathbb{R}^4$, one after another for the time-period $[T_i^-, T_i^+)$. This process repeats n times. Here C_i, $i \in \{1, 2, 3, \ldots, n\}$ represents the i^{th} chromosome of the population, T_i^- and T_i^+ start and end time of the implementation of the i^{th} chromosome. Since one

chromosome is applied after another $T_i^+ = T_{i+1}^-$.

c) Evaluate: The objective function or the cost function determines how suitable is a chromosome for the closed-loop system. The scale ranges from 0 to ∞, where zero associated with the optimal value. In the literature, a variety of cost functions such as mean square error (MSE), integration of time multiplied by absolute error (ITAE), integral of the absolute magnitude of the error (IAE), integral of the squared error (ISE) and root mean square error (RMSE) are utilized [14], [11]. In this article, the objective function uses RMSE defined as

$$E_i = \frac{1}{T}\int_{T_i^-}^{T_i^+} \sqrt{(e(t))^2}\mathrm{d}t,$$

where $i \in \{1, 2, 3, \ldots, n\}$ and $T = T_i^+ - T_i^-$ is the time interval for which the i^{th} chromosome is applied.

d) Select the fittest: The fittest chromosome in the present generation of the population is the one which produced minimum value for the cost function.

e) Termination criterion: In the practical implementation, the objective function never reaches the zero value. Hence, the chromosome which offers the objective function below a certain predetermined value, E_A, is considered optimal. If all of the chromosomes fail to provide optimal results, a new generation is created and evaluated.

f) Selection: This is the first stage of the reproduction process. In this juncture, some chromosomes from the present population are selected to involve in the reproduction process. Roulette wheel selection, stochastic universal sampling, normalized geometric selection, and tournament selection are the most commonly used selection process methods [15], [11]. This article uses roulette wheel selection method.

g) Crossover: This is another stage of the reproduction process, where the selected chromosomes undergo the crossing. It is carried out by 1) splitting the selected or parent chromosomes array at a random point and then 2.) the children chromosomes form by joining one parent's front end of separated array with the rear end of another.

h) Mutation: In this stage of the reproduction process, at randomly locations the values are changed. The mutation rate determines the number of mutations. Usually, it is less than 1%.

Since this algorithm uses random values for many of its operations, the optimum values may vary every time we restart the process. The next section presents the problems associated with the implementation of this algorithm.

V. Problems in the implementation

The following problems arise in the implementation of the algorithm discussed in the previous.

A. The previous chromosome's error, E_{i-1}, affects E_i

The chromosomes are applied one after another, that is $T_{i+1}^- = T_i^+$, in the standard algorithm as shown in Fig. 2. The plant and the controller in the closed-loop system shown in Fig. 1 are dynamic systems. Since dynamic systems need a settling time to stabilize the system, the error associated with the application of i^{th} chromosome affect the one with $i + 1^{\text{th}}$ chromosome. Hence the objective function, E_i, is incorrectly estimated.

Solution: In this article, authors solve this problem by employing a technique which uses the previous generation's fittest chromosome, C_F^-. It uses values of C_F^- for the parameters for a sufficient time-period T_s in order to settle down the output to a stable value before applying a chromosome.

B. Some chromosomes may cause instability

Since this algorithm relies on the randomly generated values for the parameters of the control, the closed-loop system may be unstable.

Solution: To deal with this problem it is proposed that the parameters should be switched to C_F^- when E_i reaches the maximum permitted RMSE value, E_M. The value for E_M is predetermined depending on the plant's tolerance towards the error.

C. Terminating the tuning process may lead to instability in the future

If the plant presented in the closed loop system is nonlinear, the parameters tuned for an operating condition may cause instability for at other operating conditions. A change in the frequency of the reference signal $r(t)$ is sufficient to trigger an instability in the system.

Solution: Avoiding the tuning process to terminate will solve this problem. It is proposed to start the tuning process when the moving root mean square (RMS), $M_E(t)$, raises over a predetermined limit, M_E^+. The moving average for a continuous error function is defined as

$$M_E(t) = \frac{1}{T_w}\int_{t-T_w}^{t} e(t)\mathrm{d}t,$$

where T_w is window size. The continuous moving average cannot be computed. Hence the user may use the discrete version of the moving average.

VI. Proposed algorithm

This section presents a modified version of the algorithm shown in Fig. 2 by including the solutions to the problems discussed earlier.

As suggested in the previous section, the application and evaluation blocks are not separable. Fig. 3 depicts the modified version of application and evaluation blocks. Fig. 4 shows the complete algorithm.

In the next following sections simulation and experimental results are shown using this algorithm to a closed loop system.

VII. Simulation results

A. Simulation setup

Simulation is performed using Matlab Simulink. A closed loop system with the plant being

$$\frac{y(s)}{u(s)} = \frac{s+3}{8s^2+5s+2}$$

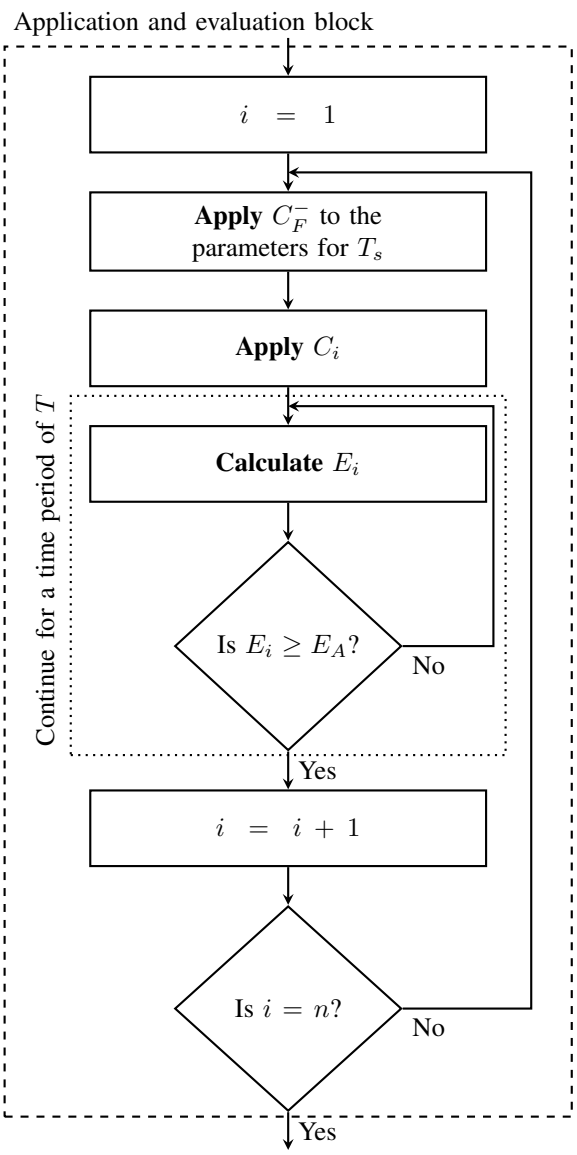

Fig. 3. Application and evaluation block, which is a part of the proposed algorithm given in Fig. 4

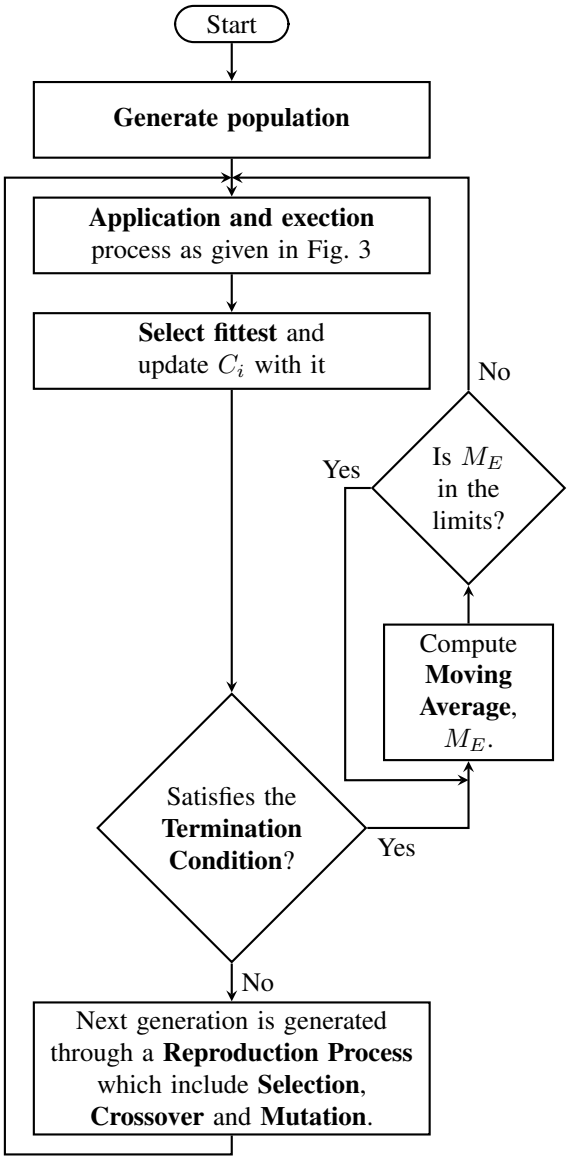

Fig. 4. Proposed algorithm

is simulated. The parameters of the system are changed at $t = 20\,\mathrm{s}$ seconds to obtain a new plant with dynamics

$$\frac{y(s)}{u(s)} = \frac{s+1}{2s^2+s+3}.$$

The constants used for the algorithm are $n = 10$, $T_s = 1\,\mathrm{s}$, $T = 0.1\,\mathrm{s}$, $E_M = 5$, $E_A = 0.1$, $M_E^+ = 0.1$, and $T_w = 0.2\,\mathrm{s}$. The initial values used are $C_F^- = [10, 10, 0, 300]^\intercal$. The constants associated to the reproduction process, crossover rate and mutation rate are taken as 0.25 and 0.1 respectively. The Simulink model is set to execute the control algorithm in continuous mode with the fixed step and the sampling time set to $1\,\mathrm{ms}$.

B. Results and discussion

Fig. 5, 6 and 7 present the simulation results. Fig. 5 shows the error, e, and the moving RMSE M_E. Fig. 6 and 7 depict the cost function E_i and the parameters used for the PID controller respectively.

As shown in Fig. 7, at $t = 0\,\mathrm{s}$ a chromosomes C_i are applied to the system one after another. At the end of the first cycle, $t = 13\,\mathrm{s}$, various optimal parameter values are identified to satisfy the algorithm's termination criterion, see Fig. 6. $C_3 = [4377.526, 841.006, 1.223, 1017.045]^\intercal$ is recognized as the fittest.

Change in the plant parameters $t = 20\,\mathrm{s}$ caused the moving average raise above the allowed $M_A^+ = 0.1$. Fig. 5 depicts the increase in e and M_A. Fig. 7 shows that the algorithm restarted the tuning process by started applying chromo-

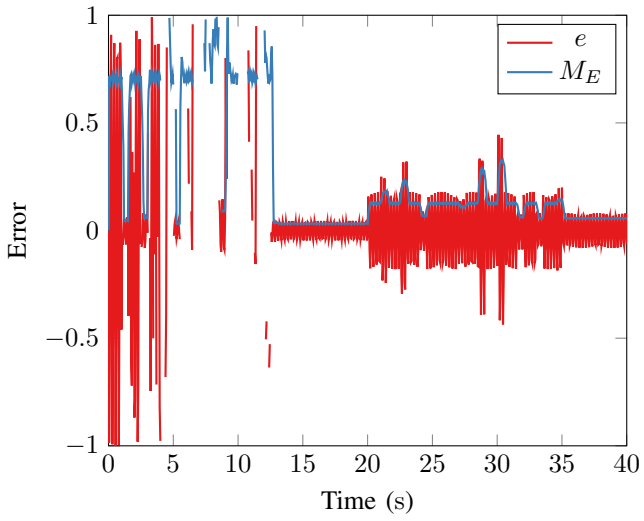

Fig. 5. Simulation results showing error and moving RMS error for closed loop system whose plant is modified at $t = 20\,\mathrm{s}$.

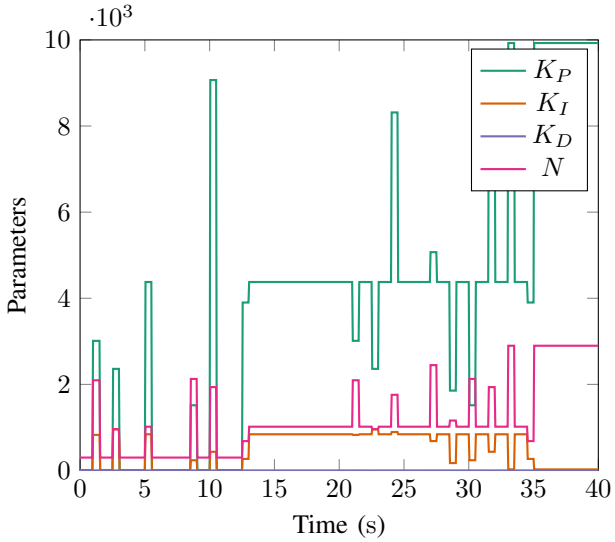

Fig. 7. Simulation results showing parameter tuning by Genetic algorithm

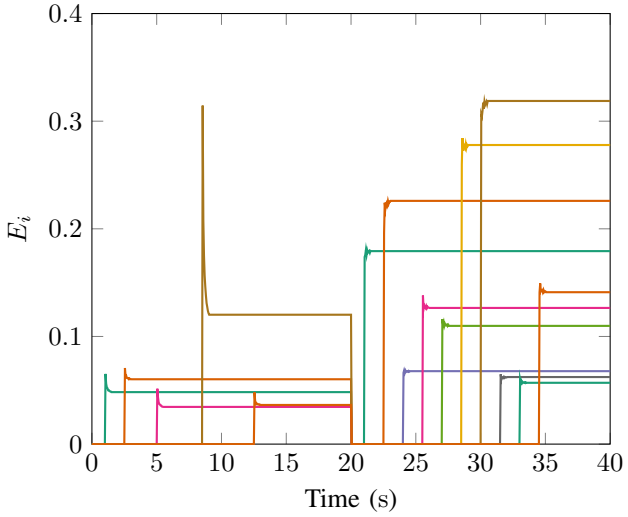

Fig. 6. Simulation results showing the values for the objective function, E_i, $i \in \{1, 2, 3, \ldots, 10\}$

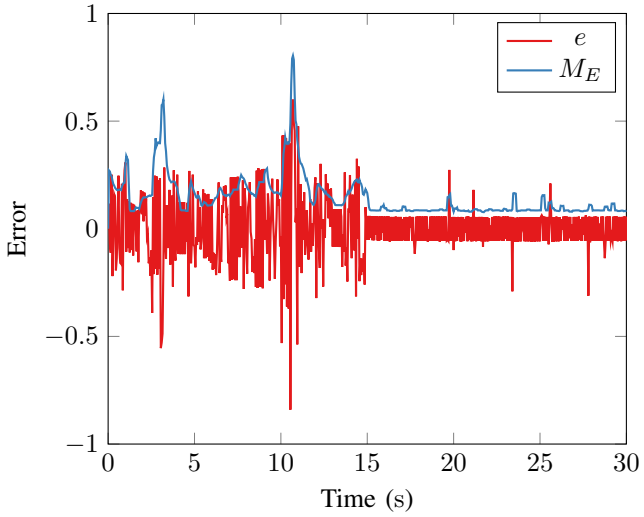

Fig. 8. Experiment results showing error and moving RMS error for a closed loop system controlling a brushless servomotor in torque mode

somes C_i. At the end of the first cycle of the application, $t = 35\,\mathrm{s}$, various optimal parameter values are identified to satisfy the algorithm's termination criterion, see Fig. 6. $C_9 = [9927.499, 25.062, 0.126, 2897.279]^\intercal$ is recognized as the fittest.

VIII. Experimental Results

A. Experimental setup

The proposed algorithm is used to control the position of a brushless motor. This experiment uses Parker's brushless servomotor SM231BE-FPSV along with its driver AR-13AE. The configuration is set to run the driver in the torque mode. The National Instrument's data acquisition (DAQ) card PCIe-6363 is used to communicate between a desktop computer with Matlab Simulink and the driver AR-13AE. The DAQ generates a 10v analog signal to control the motor's torque. The encoder coupled to the motor shaft produces 1000 pulses per revolution (PPR). It connects to the driver, and the driver generates the standard encoder output signals A, B, and Z. The Simulink is configured to read these encoder signals and send analog signals to the driver through the DAQ with the help of Real-Time Desktop Toolbox.

The constants used for the algorithm are $n = 10$, $T_s = 1\,\mathrm{s}$, $T = 0.5\,\mathrm{s}$, $E_M = 5$, $E_A = 0.25$, $M_E^+ = 0.5$, and $T_w = 0.2\,\mathrm{s}$. The initial values used are $C_F^- = [250, 90, 0.1, 1800]^\intercal$. The constants associated to the reproduction process, crossover rate and mutation rate are taken as 0.25 and 0.1 respectively. The Simulink model is set to execute the control algorithm in continuous mode with the fixed step and the sampling time set to $1\,\mathrm{ms}$.

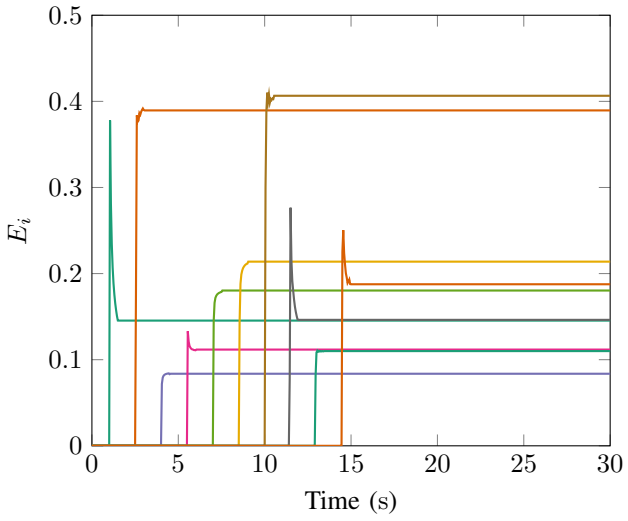

Fig. 9. Experimental results showing the values for the objective function, E_i, $i \in \{1, 2, 3, \ldots, 10\}$

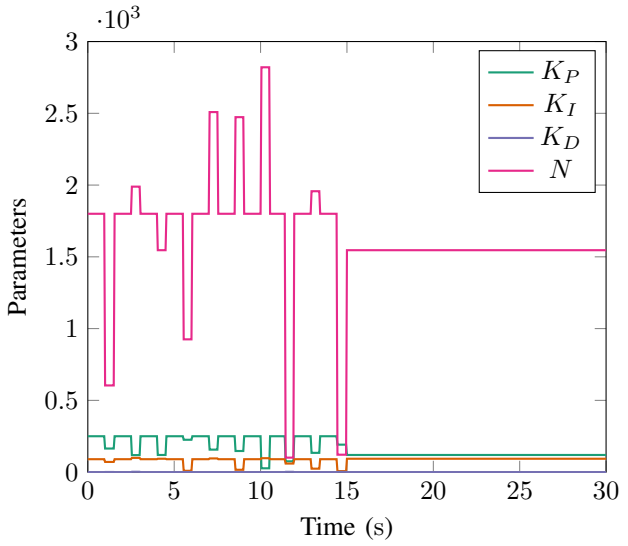

Fig. 10. Experiment results showing parameter tuning by Genetic algorithm

B. Results and discussion

Fig. 8, 9, and 10 show the experimental results. Figure 1 shows the error, e, and the moving RMSE M_E. Figure 2 and 3 depict the cost function E_i and the parameters used for the PID controller respectively.

As shown in Fig. 10, at $t = 0\,\mathrm{s}$ a chromosomes C_i are applied to the system one after another. At the end of the first cycle, $t = 15\,\mathrm{s}$, various optimal parameter values are identified to satisfy the algorithm's termination criterion, see Fig. 9. $C_3 = [119.482, 93.220, 0.230, 1546.220]^\intercal$ is recognized as the fittest.

IX. Conclusions

The following problems associated with the implementation of the standard genetic algorithm based PID tuning algorithm are identified.

1) Estimation errors caused due to other chromosomes,
2) Instability due to incompatible chromosomes, and
3) A set of tuned parameters may cause inferior performance to an inconsistent operating point.

A possible solution is provided to deal with the said problems by introducing new variables/controls into the algorithm such as E_M, M_E, M_E^+, and T_s. This algorithm is verified through simulations and experiments. It is observed that the implemented algorithm tunes the PID control parameters K_P, K_I, K_D and N. Also, it is verified that the algorithm responds to the changes in the system's operating point by reconfiguring the tunable parameters.

X. Acknowledgment

This material is based upon work supported by SEP, México under Grant No. UACOAH-PTC-383; and carried out in the Electric Lab at FIME, UAdeC, Torreón, Mexico.

References

[1] G. Ellis, *Control system design guide*, 3rd ed. San Diego, California, USA: Elsevier Academic Press, 2004.
[2] K. H. Ang, G. Chong, and Y. Li, "Pid control system analysis, design, and technology," *IEEE transactions on control systems technology*, vol. 13, no. 4, pp. 559–576, 2005.
[3] A. O'Dwyer, "Pid compensation of time delayed processes 1998-2002: a survey," in *Proceedings of the American Control Conference*. Dublin Institute of Technology, 2003, pp. 1494–1499.
[4] H. K. Khalil, *Nonlinear systems*, 3rd ed. Upper Saddle River, New Jersey, USA: Prentice Hall, 2002.
[5] Q.-C. Zhong, *Robust Control of Time-delay Systems*. Heidelberg, Germani: Springer-Verlag, 2006.
[6] A. O'Dwyer, *Handbook of PI and PID controller tuning rules*, 3rd ed. Covent Garden, London: Imperial College Press, 2009.
[7] S.-Z. He, S. Tan, F.-L. Xu, and P.-Z. Wang, "Fuzzy self-tuning of PID controllers," *Fuzzy sets and systems*, vol. 56, no. 1, pp. 37–46, 1993.
[8] S. Skogestad, "Simple analytic rules for model reduction and PID controller tuning," *Journal of process control*, vol. 13, no. 4, pp. 291–309, 2003.
[9] W.-D. Chang, R.-C. Hwang, and J.-G. Hsieh, "A self-tuning PID control for a class of nonlinear systems based on the Lyapunov approach," *Journal of Process Control*, vol. 12, no. 2, pp. 233–242, 2002.
[10] J.-C. Shen, "Fuzzy neural networks for tuning PID controller for plants with underdamped responses," *IEEE Transactions on Fuzzy Systems*, vol. 9, no. 2, pp. 333–342, 2001.
[11] J.-S. Kim, J.-H. Kim, J.-M. Park, S.-M. Park, W.-Y. Choe, and H. Heo, "Auto tuning pid controller based on improved genetic algorithm for reverse osmosis plant," *World Academy of Science, Engineering and Technology*, vol. 47, no. 2, pp. 384–389, 2008.
[12] A. Zemmit, S. Messalti, and A. Harrag, "A new improved DTC of doubly fed induction machine using GA-based PI controller," *Ain Shams Engineering Journal*, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S2090447917300205
[13] Mathworks. Take derivatives of a signal. (Accessed on 07/21/2017). [Online]. Available: http://www.mathworks.com/help/signal/ug/take-derivatives-of-a-signal.html
[14] Z.-L. Gaing, "A particle swarm optimization approach for optimum design of pid controller in avr system," *IEEE Transactions on Energy Conversion*, vol. 19, no. 2, pp. 384–391, June 2004.
[15] M. Gen and R. Cheng, *Genetic algorithms and engineering optimization*. John Wiley & Sons, 2000, vol. 7.