Proceedings of the IASTED International Conference
MODELLING, IDENTIFICATION, AND CONTROL
February 18-21, 2002, Innsbruck, Austria

DESIGN OF PID CONTROLLERS USING THE PARTICLE SWARM ALGORITHM

P. B. DE MOURA OLIVEIRA, J. BOAVENTURA CUNHA and J. PAULO COELHO
Departamento de Engenharias
Universidade de Trás-os-Montes e Alto Douro
5000 Vila Real
Portugal
Tel: +351 59 350339 Fax: + 351 59 320480
e-mail: oliveira@utad.pt

ABSTRACT

Particle swarm optimisation is proposed as an alternative technique to the controller design for single-input single-output systems. The evolutionary scheme proposed is applied to design the most popular controller within the process control industry: the proportional, integral and derivative (PID) controller. The particle swarm optimisation algorithm is applied to the design of both linear and non-linear PID controllers. In the non-linear control configuration a feedforward neural network is deployed to generate a non-linear gain that acts as a filter to the tracking error inputted to the PID controller. This evolutionary approach is illustrated by simulation examples in which the PID is used to control a delayed second order system.

KEYWORDS: PID control, Particle Swarm Optimisation, Neural Networks.

1. INTRODUCTION

The importance and role of PID controllers in the past of feedback control systems is well reported by Bennett [2]. Despite major developments of advanced control techniques over the last thirty years, the most popular controller used in industrial processes is of the PID type. This is because of its simple structure and reliability in a wide range of operating conditions. Some important considerations about the possible future of PID controllers are stated by Åström and Hägglund [1].

Evolutionary techniques such as genetic algorithms (GAs) [3] [8], and population based incremental learning (PBIL) [5] have proved to be excellent optimisation tools to design and tune linear PID controllers [6] and non-linear PID controllers [7] [4]. Despite of the success that GAs had attained in solving complex optimisation engineering problems, namely in the computed aided control system design during the last ten years, there has been considerable research effort directed into developing new optimisation algorithms inspired on natural concepts. One of the most recently introduced algorithms is the particle swarm optimisation algorithm [10].

This paper reports the use of the particle swarm optimisation algorithm to the design of linear PID and non-linear PID controllers. In the latter, a feedforward neural network is used to generate a non-linear gain that will act on the set-point tracking error, prior to the controller. The paper is divided into five sections. Following the introduction the second section provides an overview of the particle swarm optimisation algorithm. In section three the PSO algorithm is proposed an alternative evolutionary design technique both to linear and non-linear neural based PID control configurations. Section four shows simulation results obtained for a second order model with a time delay. In section five some conclusions are drawn.

2. PARTICLE SWARM OPTIMISATION ALGORITHM: AN OVERVIEW

The particle swarm optimisation (PSO) algorithm was proposed by Kennedy and Eberhart [9], [10] and it is conceptually based on the social behaviour of groups of organisms such as herds, schools and flocks. As an evolutionary based technique PSO it is a population (swarm) based algorithm, formed by a set of particles, which represent potential solutions for a given problem. Each particle of a set of swarm size m, moves through a n-dimensional search space (as birds in a flock), with an associated position vector $X_i(t) = (x_{i1}(t), x_{i2}(t), \ldots, x_{in}(t))$ and velocity vector $V_i(t) = (v_{i1}(t), v_{i2}(t), \ldots, v_{in}(t))$ for the current evolutionary iteration t.

The original PSO model integrates two types of knowledge acquisition by a particle: through it's own experience and from social sharing from other population members. The former was termed *Cognition-Only Model* and the latest *Social-Only Model* [10]. The behaviour of each particle is based on these two types of knowledge and their current position regarding the search. Kennedy modelled particle behaviour by using the following two equations:

$$v_{id}(t+1)=v_{id}(t)+\varphi_1.(p_{id}(t)-x_{id}(t))+\varphi_2.(p_{gd}(t)-x_{id}(t)) \quad (1)$$

$$x_{id}(t+1)=x_{id}(t)+v_{id}(t+1) \quad (2)$$

in which d represents the dimension index, $1 \le d \le n$, $p_{id}(t)$ represents the best previous position of particle i at the current iteration t, $p_{gd}(t)$ represents the global best, in the current iteration, for a pre-defined neighbourhood type. Parameter φ_1 is known as the *Cognitive Constant* and φ_2 as the *Social Constant* and represent uniformly distributed random numbers generated in a pre-defined interval.

An additional parameter was incorporated into equation (1) [16] resulting in equation (3):

$$v_{id}(t+1)=\omega v_{id}(t)+\varphi_1.(p_{id}(t)-x_{id}(t))+\varphi_2.(p_{gd}(t)-x_{id}(t)) \quad (3)$$

in which ω was termed *Inertia Weight*. The value given to the inertia weight will affect the type of search in the following way: a large inertia weight will direct the PSO for a global search while a small inertia weight will direct the PSO for a local search. This parameter can vary linearly from a larger value to a smaller value in order to make the search global in the early run and local in the end of the run. The constants φ_1, φ_2 and ω can be interpreted as the confidence that each particle has in its current position, its own experience and its neighbours experience, respectively.

The neighbourhood can be of different size and topology. Each particle can take into account either:
(i) the social information from a list of particles pre-defined in the beginning of the simulated evolution. This list can incorporate all the population individuals, with an individual being able to use the best solution found by every other member. This full-connected social network structure was termed *Star*. In other list definitions, an individual uses only k adjacent neighbours organised in a *Circle* and *Wheel* topology [11].
(ii) the physical information which considers distance between neighbour individual evaluated using some metric definition.

A simplified version of equation (1) was proposed by Clerc and Kennedy [11] by considering $\varphi_1=\varphi_2=\varphi$ and defining an intermediate position p_{ig} between the best previous position p_i and the global best p_g defined by equation (4):

$$p_{igd}(t)=p_{id}(t)\frac{p_{id}(t)-p_{gd}(t)}{2} \quad (4)$$

resulting in a modified velocity governed by equation (5):

$$v_{id}(t+1)=v_{id}(t)+\varphi.(p_{igd}(t)-x_{id}(t)) \quad (5)$$

Clerc [20] proposed the use of a constriction coefficient χ incorporated in the simplified velocity equation by:

$$v_{id}(t+1)=\chi(v_{id}(t)+\varphi.(p_{igd}(t)-x_{id}(t))) \quad (6)$$

with $\varphi \ge 4$. The constriction coefficient can be evaluated by using the following equation:

$$\chi=\frac{2k}{\left|2-\varphi-\sqrt{\varphi^2-4\varphi}\right|} \quad 0 \le k \le 1 \quad (7)$$

The effect of this coefficient is to promote convergence over time. The parameters $k=1$ and $\varphi=4.1$ are suggested [11] as good values. Another constriction method version results in the following velocity equation modification (3):

$$v_{id}(t+1)=\chi[\omega v_{id}(t)+\varphi_1.(p_{id}(t)-x_{id}(t))+\varphi_2.(p_{gd}(t)-x_{id}(t))] \quad (8)$$

The velocity is limited by a maximum, V_{max}, meaning the maximum jump that each particle can make in each iteration. The selected value for V_{max} should not be too high to avoid oscillations, or too low to avoid search traps. The inertia weight and maximum velocity parameters selection in the PSO algorithm was studied and reported by Shi and Eberhart [17]. Each particle position should also be located within its dynamic range *[-Xmax,Xmax]*. Evolutionary notions of breeding and the use of subpopulations were incorporated into the PSO by Løvberg et. al [19] resulting in a Hybrid PSO. Another modification of the PSO algorithm was proposed by Parsopoulos and Vrahatis [18] in order to make the algorithm to locate all global minima of a function, by using a "stretching" technique.

3. CONTROLLER DESIGN USING THE PARTICLE SWARM ALGORITHM

3.1 DESIGN OF LINEAR PID CONTROLLERS

It is well known that the PID controller is governed in the continuous domain by equation (9):

$$u(t)=K_p\left[e(t)+\frac{1}{T_i}\int_o^t e(t)dt+T_d\frac{de(t)}{dt}\right] \quad (9)$$

in which, K_p represents the proportional gain, T_i and T_d the integral and derivative time constants, respectively. A digital implementation of the continuous PID controller represented by equation (9) can be deployed in either absolute or incremental form. The absolute algorithm with an anti-windup scheme is adopted in this study. Digital PID controllers can be governed on the discrete time set by control law equations of the absolute form:

$$u(kT)=K_p e(kT)+K_i z(kT)+K_d\,\Delta e(kT) \quad (10)$$

in which u is the controller output, K_i is the integral gain, K_d is the derivative gain, T is the sampling period and:

$$e(kT)=v(kT)-y(kT) \quad (11)$$

$$\Delta e(kT)=e(kT)-e((k-1)T) \quad (12)$$

$$z(kT)=z((k-1)T)+Te(kT) \quad (13)$$

are the error, first backward error difference, the integral of the error discrete approximation, respectively and v is the reference signal.

In order to use the PSO algorithm to optimise a control system with a PID controller it is necessary to encode the tuning parameters. Thus each population member represents the proportional, integral and derivative parameters, by using a real-based coding scheme. The PID controller design is accomplished by optimising the system response to a unit set-point change by minimising a time-domain performance criterion such as the *Integral of the Square Error (ISE)*.

3.2 DESIGN OF NON-LINEAR PID CONTROLLERS WITH NEURAL NETWORKS

The non-linear PID control configuration studied in this research is shown in Figure 1, where k is a non-linear gain, $Gc(z)$ is the transfer function of a linear PID controller and $Gp(z)$ is the process transfer function.

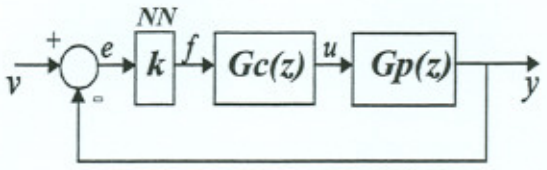

Figure 1: Non-linear control system.

For the non-linear gain k Seraji [21] proposed the use of either a sigmoidal function or a hyperbolic function. The advantage of this type of control structure is that when magnitude of the error is large the non-linear gain increases the error inputted to the controller, originating a large correction, so the system has a fast response time. When the magnitude of the error is small the non-linear gain decreases the error inputted to the controller preventing the occurrence of a large overshoot. An alternative design technique for this non-linear control configuration was proposed by Oliveira et al. [22], in which the PBIL algorithm was used to select an Lagrangian interpolated profile both for the non-linear gain and PID gains.

Two artificial feedforward neural networks architectures illustrated in Figure 2 are proposed to generate the non-linear gain k.

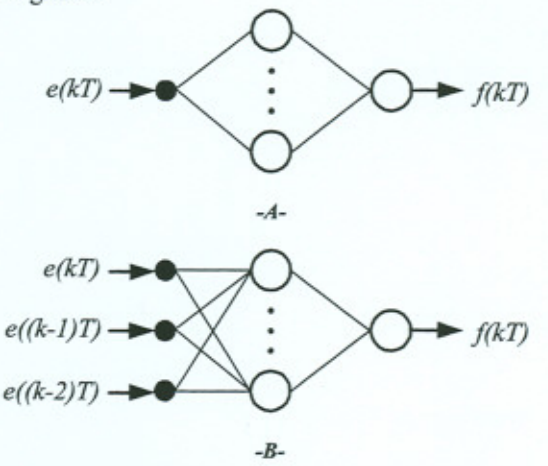

Figure 2: Feedforward neural networks configurations proposed as the non-linear gain.

Configuration A is governed by equations (14) and (15):

$$net_j(kT)=w_j^1\,e(kT)+bh_j \quad (14)$$

$$f(kT)=\sum_{i=0}^{n_h} w_j^2\,S_j\left(net_j(kT)\right)+b \quad (15)$$

where: net_j is the input of the j hidden neuron, w_j^1 is the weight connecting the input unit to hidden neuron j, bh_j is the bias of hidden neuron j, f is the output of the neural network, n_h is the number of neurons in the hidden layer, w_j^2 is the weight connecting the j hidden neuron to the output unit, S_j is the activation function of j neuron and b is the output bias.

Configuration B is governed by equations (16) and (17):

$$net_j(kT) = \sum_{i=0}^{n_i} w_{j,i+1}^1 \, e((k-i)T) + bh_j \quad (16)$$

$$f(kT) = \sum_{i=0}^{n_h} w_j^2 \, S_j(net_j(kT)) + b \quad (17)$$

where: net_j is the input of the j hidden neuron, n_i is the number of input units, $w_{j,i}^1$ is the weight connecting the input unit i to hidden neuron j, bh_j is the bias of hidden neuron j, f is the output of the neural network, n_h is the number of neurons in the hidden layer, w_j^2 is the weight connecting the j hidden neuron to the output unit, S_j is the activation function of j neuron and b is the output bias. Both networks topologies have one hidden layer. The activation function used in this work was a sigmoid function for the hidden layer and a linear function in the output layer.

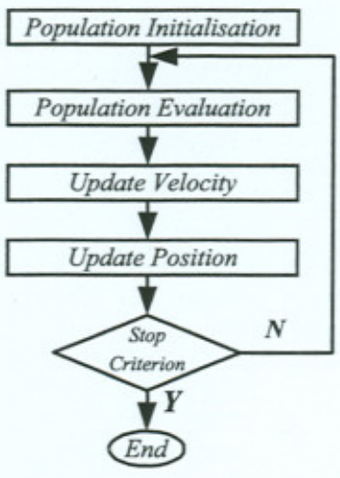

Figure 3: Basic PSO algorithm.

In order to use the PSO algorithm to design the non-linear PID with four neurons in one hidden layer, each particle in the population is represented by vector (18) and (19) for topology A and B, respectively:

$$\left(w_1^1, \cdots, w_4^1, bh_1, \cdots, bh_4, w_1^2 \cdots w_4^2, b, K_p, K_i, K_d\right) \quad (18)$$

$$\left(\begin{matrix} w_{1,1}^1, \cdots, w_{1,3}^1, w_{2,1}^1, \cdots, w_{2,3}^1, w_{3,1}^1, \cdots, w_{3,3}^1, \\ w_{4,1}^1, \cdots, w_{4,3}^1, bh_1, \cdots, bh_4, w_1^2 \cdots w_4^2, b, K_p, K_i, K_d \end{matrix}\right) \quad (19)$$

and are evolved, in this case, from a randomly initialised population. The simple PSO algorithm loop can be illustrated by Figure 3, in which the particles velocities and positions are updated using equations (2) and (3), respectively.

4. SIMULATION RESULTS

Let's consider the control system illustrated in Figure 1. The process to be controlled can be modelled by a second order model with the addition of a time delay described in the discrete time domain by (20). The sampling time used was $T=0.01$ seconds.

$$G_p(z) = \frac{0.0012091 z^{-9} + 0.001169 z^{-10}}{z^2 - 1.9025 z + 0.90484} \quad (20)$$

Two examples are presented in the next two sub-sections using the PSO algorithm as the design tool for linear and non-linear discrete PID controllers. The objective of these examples is to design a PID controller by optimising the system response to a unit set-point change. The optimisation criterion used is the discrete *ISE*. The population size used in the PSO algorithm is $n=50$, a constriction coefficient of $\chi=1$. The inertia weight ω is set to change linearly in the interval [0.7,0.4] during the evolution through 100 epochs. Parameters φ_1 and φ_2 are uniformly distributed random numbers generated in the interval [0,1]. The adjustments of the velocity position vectors are done using expressions (2) and (3).

4.1 EXAMPLE 1: LINEAR PID CONTROLLER

The result obtained by designing a linear PID controller using the PSO algorithm, by optimising the parameters $\{K_p, K_i, K_d\}$ in the interval [0,5], is shown in Figure 4 for an unit step response.

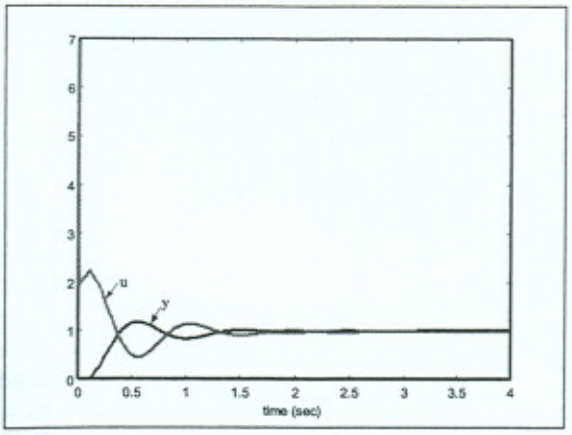

Figure 4: Unit step response for the linear PID.

The controller output was allowed to vary in the interval [-8,8]. The optimised parameters were: $\{K_p$=1.89, K_i=3.02, K_d=5.00 $\}$, with an *ISE=24,6*.

4.2 EXAMPLE 2: NON-LINEAR NEURAL PID CONTROLLER

The result obtained by designing a non-linear PID controller using the PSO algorithm to optimise the parameters in vector (18) corresponding to the weights of the neural network with configuration *A* and the PID gains, is shown in Figure 5 for an unit step response.

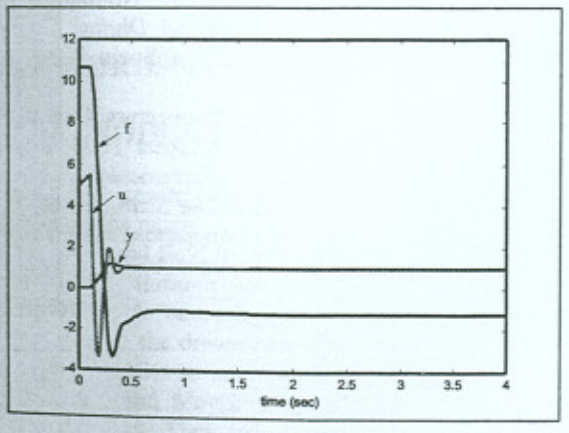

Figure 5: Unit step response for the non-linear neural PID. Topology A.

The weights and bias ranges used were: $-6 \leq w_j^1, w_j^2 \leq +6$; $-2 \leq bh_j$, $b \leq 2$ with $1 \leq j \leq 4$ and $0 \leq K_p, K_i, K_d \leq 5$. The controller output was allowed to vary in the interval [-8,8]. The optimised parameters are presented in Table 1 and the *ISE=19,04*.

Table 1: Optimised parameters for non-linear PID controller using the neural network topology A.

w_1^1	w_2^1	w_3^1	w_4^1	bh_1	bh_2	bh_3	bh_4
2.60	5.90	-1.09	-4.64	-2.00	1.47	-0.45	-0.85
w_1^2	w_2^2	w_3^2	w_4^2	b	K_p	K_i	K_d
6.00	-1.23	-4.33	-2.67	2.00	0.47	0.53	4.95

The result obtained by designing a non-linear PID controller using the PSO algorithm to optimise the parameters represented by vector (19) corresponding to the neural network weights with configuration *B* and the PID gains, is shown in Figure 6 for an unit step response. The weights and bias range were: $-6 \leq w_{j,i}^1, w_j^2 \leq +6$; $-2 \leq bh_j$, $b \leq 2$ with $1 \leq j \leq 4$, $1 \leq i \leq 3$; $0 \leq K_p, K_i, K_d \leq 5$. The controller output was allowed to vary in the interval [-8,8]. The optimised parameters are presented in Table 2 with an *ISE=18,9*.

The results indicate that both non-linear PID controllers improve significantly the performance obtained compared with the linear case, as it was expected. The used of neural network topology B did not improve the performance significantly compared with topology A.

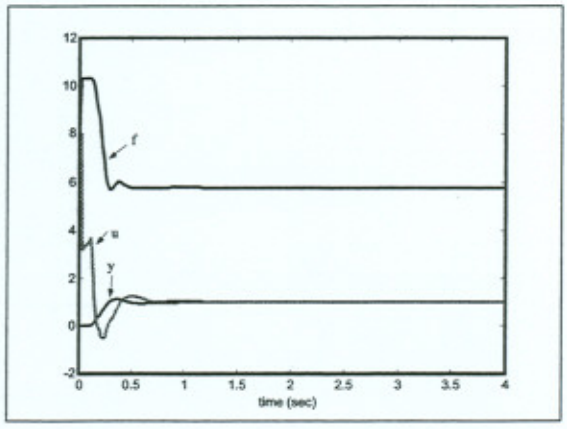

Figure 6: Unit step response for the non-linear PID. Topology B.

Table 2: Optimised parameters for non-linear PID controller using the neural network topology B.

$w_{1,1}^1$	$w_{1,2}^1$	$w_{1,3}^1$	bh_1	bh_2	bh_3	bh_4	$w_{2,1}^1$	$w_{2,2}^1$
-3.56	5.57	3.48	-2.00	1.44	-1.03	-2.0	1.27	-3.19
$w_{2,3}^1$	$w_{3,1}^1$	$w_{3,2}^1$	$w_{3,3}^1$	$w_{4,1}^1$	$w_{4,2}^1$	$w_{4,3}^1$	w_1^2	w_2^2
-4.04	-5.96	1.74	0.16	-0.25	6.00	0.52	5.99	3.24
w_3^2	w_4^2	b	K_p	K_i	K_d			
0.94	6.00	1.47	0.31	0.59	3.64			

5. CONCLUSION

The particle swarm optimisation algorithm was proposed as an alternative technique to the controller design for single-input single-output systems. The evolutionary scheme proposed is applied to the design of the most popular controller within the process control industry: the PID controller. The particle swarm optimisation algorithm was applied to design both linear and non-linear PID controllers. In the non-linear control configuration a feedforward neural network is deployed to generate a non-linear gain that acts as a filter to the tracking error inputted to the PID controller. Simulation results show that the PSO algorithm is a good alternative to other evolutionary design techniques, such as the genetic algorithm, because of its good convergence rate and simplicity of implementation. The PSO algorithm proved to be able to design both linear and non-linear neural based PID controllers, with the latest requiring the optimisation of a significant number of parameters.

REFERENCES

[1] Åström K. J. and Hägglund T., The Future of PID Control, *IFAC Workshop on Digital Control: Past, present and future*, Terrassa, Spain, 2000, 19-30.

[2] Bennet S., The Past of PID Controllers, *IFAC Workshop on Digital Control: Past, present and future*, Terrassa, Spain, 2000, 3-13.

[3] Holland J. H., Adaptation in Natural and Artificial Systems, 1st MIT Press ed., 1975.

[4] De Moura Oliveira P. B and Solteiro Pires E J Projecto de Controladores PID Discretos Nao-Lineares Usando PBIL, *XX Jornadas de Automatica*, Salamanca, Spain, 1999, 229-234.

[5] Baluja S., Population Based Incremental Learning: A Method for Integrating Genetic search Based Function Optimization and Competitive Learning, Technical report CMU-CS-95-163, School of Computer Science, Carnegie Melon University, USA, 1994.

[6] De Moura Oliveira and Jones A. H., Genetic Design of Two Degrees-of-Freedom PID Controllers for Set-Point Tracking and Disturbance Rejection, *3rd Portuguese Conference on Automatic Control*, Coimbra, Portugal, 1998, 111-116.

[7] Jones A H, Genetic Tuning of Neural Non-Linear PID Controllers, Artificial Neural Nets and Genetic Algorithms, Pearson *et al.* Editors, Springer Verlag, 1995, 412-415.

[8] Goldberg E D, Genetic Algorithms in Search, Optimization and Machine Learning, Adison Wesley P.C., 1989.

[9] Kennedy J. and Eberhart R. C., Particle Swarm Optimization, *Proceedings of the 1995 IEEE Int. Conference on Neural Networks*, Perth, Australia, 1995, 1942-1948.

[10] Kennedy J., The Particle Swarm: Social Adaptation of Knowledge, *Proceedings of the 1997 IEEE Int. Conference on Evolutionary Computation*, Indianapolis, US, 1997, 303-308.

[11] Kennedy J. and Eberhart R. C., Swarm Intelligence, Academic Press, 2001, 344-345.

[16] Shi Y. and Eberhart R. C., Empirical Study of Particle Swarm Optimization, *Proceedings of the 1999 IEEE Congress of Evolutionary Computation*, 3, 1999, 1945-1950.

[17] Shi Y. and Eberhart R. C., Parameter Selection in Particle Swarm Optimisation, Evolutionary Programming VII, Lecture Notes in Computer Science, 1447, 1998, 591-600, Springer Verlag.

[18] Parsopoulos K E and Vrahatis M N, Modification of the Particle Swarm Optimizer for Locating All the Global Minima, *5th International Conference on Neural Networks and Genetic Algorithms (ICANNGA)*, Springer-Computer Science, 2001, 324-327.

[19] Løvberg M., Rasmussen T. K. and Krink T., Hybrid Particle Swarm Optimiser with Breeding and Subpopulations, *Proceedings of the third Genetic and Evolutionary Computation Conference (GECCO-2001)*, 2001.

[20] Clerc M., The Swarm and the Queen Towards a Deterministic and Adaptive Particle Swarm Optimization, *Proceedings of the 1999 Congress of Evolutionary Computation*, IEEE Press, 3, 1999, 1951-1957.

[21] Seraji H., "A New Class of Nonlinear PID Controllers", *IFAC Workshop on Digital Control: Past, present and future*, Terrassa, Spain, 2000, 540-545.

[22] De Moura Oliveira P. B., Boaventura Cunha J. e Cordeiro M., Design of Non-Linear PI Controllers using Population Based Incremental Learning Algorithm", *ICARCV 2000: The Sixth International Conference on Control, Automation, Robotics and Vision*, CDROM Ed., Abstract 60-61, Singapore, 2000.