Computers and Mathematics with Applications 60 (2010) 1088–1104

Contents lists available at ScienceDirect

Computers and Mathematics with Applications

journal homepage: www.elsevier.com/locate/camwa

Chaos driven evolutionary algorithms for the task of PID control

Donald Davendra [a,*], Ivan Zelinka [b,a], Roman Senkerik [a]

[a] Tomas Bata University in Zlin, Faculty of Applied Informatics, Department of Informatics and Artificial Intelligence, Nad Stranemi 4511, Zlin 76001, Czech Republic

[b] Faculty of Electrical Engineering and Computing Science, Technical University of Ostrava, Tr. 17. Listopadu 15, Ostrava, Czech Republic

ARTICLE INFO

Keywords:
Chaos maps
Differential Evolution
Self-organizing migrating algorithm
PID controller

ABSTRACT

Chaos driven Differential Evolution algorithm and Self-Organizing Migrating Algorithm are presented in this paper for the task of PID (Proportional–Integral–Derivative) controller optimization. The dissipative chaotic Lozi map is embedded as a number generator inside DE and SOMA in order to avoid local optima stagnation and embed a superior search strategy. Three unique PID controller problems are presented and successfully resolved using these new approaches. The obtained results compare favorably with published results.

1. Introduction

A Proportional–Integral–Derivative (PID) controller is a three-term controller that has a long history in the automatic control field, starting from the beginning of the last century [1]. Owing to its intuitiveness and its relative simplicity, in addition to satisfactory performance which it is able to provide with a wide range of processes, it has become in practice the standard controller in industrial settings.

Evolutionary design and synthesis of PID controllers is a recent manifestation. Evolutionary algorithms (EAs) are considered as a powerful set of tools in the task of any complex optimization. EAs arose with the advent of computers and has now evolved into a very complex set of algorithms. Almost all conceived natural occurring phenomena have been translated into a set of evolutionary heuristics. These heuristics in turn have been applied to the task of PID controller design in its various formats. Chang [2] developed a new variant of Genetic Algorithm (GA) [3] for PID tuning, whereas Yachen and Yueming [4] developed a Simulated Annealing (SA) [5] approach. Dong [6] and Chakraborty [7] during the past few years have compared the performances of Differential Evolution (DE) and Particle Swarm Optimization (PSO) [8] in the task of PID controller design. Further application can be found in [9–11] amongst others.

This research deals with the application of EAs driven by *chaotic maps* in PID controller design. Recent research in chaos driven heuristics has been fueled with the predisposition that unlike stochastic approaches, a chaotic approach is able to bypass local optima stagnation. This one clause is of deep importance to EAs. A chaotic approach generally uses the chaotic map in the place of a random number generator [12]. This causes the heuristic to map unique regions, since the chaotic map iterates to new regions. The onus is then to select a very good chaotic map as the random generator.

Davendra and Zelinka [13] embedded chaotic maps inside DE and compared the performance of canonical and chaos mutated DE in PID controller design. The results confirmed that chaos driven DE performs better than canonical DE over a set of PID problems. This paper builds on the previous work of Davendra and Zelinka [13] with the application of chaos driven Self Organizing Migrating Algorithm (SOMA).

* Corresponding author.
E-mail addresses: davendra@fai.utb.cz (D. Davendra), zelinka@fai.utb.cz (I. Zelinka), senkerik@fai.utb.cz (R. Senkerik).

0898-1221/$ – see front matter © 2010 Elsevier Ltd. All rights reserved.
doi:10.1016/j.camwa.2010.03.066

This paper is divided into the following parts. Sections 2 and 3 introduces the basic principles of PID specifications and controller tuning. Section 4 describes DE and SOMA. Section 5 gives the mathematical description of the *Lozi Map*. Section 6 gives the experimentation results of the three attempted PID controller tuning problems and finally, in Section 7 the conclusions are presented.

2. PID controller

The PID controller contains three unique parts; proportional, integral and derivative controller [14]. The following sections gives a brief description of the different components.

2.1. Proportional algorithm

The mathematical representation is given as,

$$\frac{mv(s)}{e(s)} = k_c \tag{1}$$

in the Laplace domain or as,

$$mv(t) = mv_{ss} + k_c e(t) \tag{2}$$

in the time domain.

The proportional mode adjusts the output signal in direct proportion to the controller input (which is the error signal, **e**). The adjustable parameter to be specified is the controller gain, k_c. The larger the k_c, the more the controller output will change for a given error [14].

2.2. Proportional integral algorithm

The mathematical representation is given as,

$$\frac{mv(s)}{e(s)} = k_c \left[1 + \frac{1}{T_i s}\right] \tag{3}$$

in the Laplace domain or as,

$$mv(t) = mv_{ss} + k_c \left[e(t) + \frac{1}{T_i}\int e(t)\mathrm{d}t\right] \tag{4}$$

in the time domain.

The additional integral mode corrects for any offset (error) that may occur between the desired value (setpoint) and the process output automatically over time. The adjustable parameter to be specified is the integral time (T_i) of the controller [14].

2.3. Proportional integral derivative algorithm

The mathematical representation is given as,

$$\frac{mv(s)}{e(s)} = k_c \left[1 + \frac{1}{T_i s} + T_D s\right] \tag{5}$$

in the Laplace domain or as,

$$mv(t) = mv_{ss} + k_c \left[e(t) + \frac{1}{T_i}\int e(t)\mathrm{d}t + T_D \frac{\mathrm{d}e(t)}{\mathrm{d}t}\right]. \tag{6}$$

Derivative action anticipates where the process is heading, by looking at the time rate of change of the controlled variable (its derivative). T_D is the 'rate time' and this characterizes the derivative action (with units of minutes).

The parallel PID controller is given as in Eq. (7).

$$mv(s) = k_c e(s) + \frac{1}{T_i s} e(s) + T_D s e(s). \tag{7}$$

A further alternative simplified form is given in Eq. (8).

$$G(s) = K\left(1 + \frac{1}{sT_i} + sT_d\right). \tag{8}$$

The PID form most suitable for analytical calculations is given in Eq. (9).

$$G(s) = k + \frac{k_i}{s} + sk_d. \tag{9}$$

The parameters are related to the standard form through: $k = K$, $k_i = \frac{K}{T_i}$ and $k_d = KT_d$.

The advantage is that the parameters appear linearly and it is possible to obtain pure proportional, integral, or derivative action by finite values of the parameters.

3. Controller tuning

Controller tuning involves the selection of the best values of k_c, T_i and T_D. This is often a subjective procedure and is certainly process dependent. When tuning a PID algorithm, generally the aim is to match some preconceived 'ideal' response profile for the closed loop system. The following response profiles are typical [15].

1. *Overshoot*: this is the magnitude by which the controlled 'variable swings' past the setpoint. 5%/10% overshoot is normally acceptable for most loops.
2. *Rise time*: the time it takes for the process output to achieve the new desired value. One-third the dominant process time constant would be typical.
3. *Decay ratio*: this is the ratio of the maximum amplitude of successive oscillations.
4. *Settling time*: the time it takes for the process output to die to between, say ±5% of setpoint.

3.1. Optimization specifications

The properties of the transfer functions can also be based on integral criteria [11]. Let $e(t)$ be the error caused by reference values or disturbances and let $u(t)$ be the corresponding control signal. The performance index is calculated over a time interval; T, normally in the region of $0 \leq T \leq t_s$ where t_s is the settling time of the system. The following performance indices are of note:

3.1.1. Integral of time multiplied by absolute error (ITAE)

$$I_{\mathrm{ITAE}} = \int_0^T t\,|e(t)|\,\mathrm{d}t. \tag{10}$$

The ITAE weights the error with time and hence emphasizes the error values later on in the response rather than the initial large errors.

3.1.2. Integral of absolute magnitude of the error (IAE)

$$I_{\mathrm{IAE}} = \int_0^T |e(t)|\,\mathrm{d}t. \tag{11}$$

IAE gets the absolute value of the error to remove negative error components. IAE is good for simulation studies.

3.1.3. Integral of the square of the error (ISE)

$$I_{\mathrm{ISE}} = \int_0^T e^2(t)\mathrm{d}t. \tag{12}$$

The ISE squares the error to remove negative error components. ISE discriminates between over-damped and under damped systems, i.e. a compromise minimizes the ISE.

3.1.4. Mean of the square of the error (MSE)

$$I_{\mathrm{MSE}} = \frac{1}{n}\sum_{i=1}^{n}(e(t))^2. \tag{13}$$

MSE reflects all variation and deviation from the target value.

4. Evolutionary algorithms

4.1. Differential Evolution

DE [16] is a stochastic heuristic which uses vector differentials between solutions as a means of propagation.

In order to describe DE, a schematic is given in Fig. 1.

There are essentially five sections to the code. Section 1 describes the input to the heuristic. D is the size of the problem, $G_{\max}$ is the maximum number of generations, NP is the total number of solutions, F is the scaling factor of the solution and CR is the factor for crossover. F and CR together make the internal tuning parameters for the heuristic.

$$
\begin{array}{l}
1.\text{Input} : D, G_{\max}, NP \geq 4, F \in (0, 1+), CR \in [0, 1], \text{and initial bounds} : x^{(lo)}, x^{(hi)}. \\
2.\text{Initialize} : \left\{ \begin{array}{l} \forall i \leq NP \wedge \forall j \leq D : x_{i,j,G=0} = x_j^{(lo)} + rand_j\,[0, 1] \bullet \left(x_j^{(hi)} - x_j^{(lo)}\right) \\ i = \{1, 2, ..., NP\}, j = \{1, 2, ..., D\}, G = 0, rand_j[0, 1] \in [0, 1] \end{array} \right. \\
\left\{ \begin{array}{l} 3.\text{While} \quad G < G_{\max} \\ \left\{ \begin{array}{l} \forall i \leq NP \left\{ \begin{array}{l} 4. \quad \text{Mutate and recombine :} \\ 4.1 \quad r_1, r_2, r_3 \in \{1, 2, ..., NP\}, \\ \quad \text{randomly selected, except} : r_1 \neq r_2 \neq r_3 \neq i \\ 4.2 \quad j_{rand} \in \{1, 2, ..., D\}, \text{randomly selected once each } i \\ 4.3 \quad \forall j \leq D, u_{j,i,G+1} = \left\{ \begin{array}{l} x_{j,r_3,G} + F \cdot (x_{j,r_1,G} - x_{j,r_2,G}) \\ \text{if} \quad (rand_j[0, 1] < CR \vee j = j_{rand}) \\ x_{j,i,G} \quad \text{otherwise} \end{array} \right. \\ 5. \quad \text{Select} \\ x_{i,G+1} = \left\{ \begin{array}{l} u_{i,G+1} \ \text{if} \quad f(u_{i,G+1}) \leq f(x_{i,G}) \\ x_{i,G} \quad \text{otherwise} \end{array} \right. \end{array} \right. \\ G = G + 1 \end{array} \right. \end{array} \right.
\end{array}
$$

Fig. 1. Differential Evolution algorithm.

Section 2 outlines the initialization of the heuristic. Each solution $x_{i,j,G=0}$ is created randomly between the two bounds $x^{(lo)}$ and $x^{(hi)}$. The parameter j represents the index to the values within the solution and i indexes the solutions within the population. So, to illustrate, $x_{4,2,0}$ represents the second value of the fourth solution at the initial generation.

After initialization, the population is subjected to repeated iterations in Section 3.

Section 4 describes the conversion routines of DE. Initially, three random numbers r_1, r_2, r_3 are selected, unique to each other and to the current indexed solution i in the population in 4.1. Henceforth, a new index j_{rand} is selected in the solution. j_{rand} points to the value being modified in the solution as given in 4.2. In 4.3, two solutions, $x_{j,r_1,G}$ and $x_{j,r_2,G}$ are selected through the index r_1 and r_2 and their values subtracted. This value is then multiplied by F, the predefined scaling factor. This is added to the value indexed by r_3.

However, this solution is not arbitrarily accepted in the solution. A new random number is generated, and if this random number is less than the value of *CR*, then the new value replaces the old value in the current solution. Once all the values in the solution are obtained, the new solution is vetted for its fitness or value and if this improves on the value of the previous solution, the new solution replaces the previous solution in the population. Hence the competition is only between the new *child* solution and its *parent* solution.

Price [16] has suggested ten different working strategies. It mainly depends on the problem on hand for which strategy to choose. The strategies vary on the solutions to be perturbed, number of differing solutions considered for perturbation, and finally the type of crossover used. The following are the different strategies being applied.

1: DE/best/1/exp:	$u_{i,G+1} = x_{\text{best},G} + F \bullet \left(x_{r_1,G} - x_{r_2,G}\right)$
2: DE/rand/1/exp:	$u_{i,G+1} = x_{r_1,G} + F \bullet \left(x_{r_2,G} - x_{r_3,G}\right)$
3: DE/rand-best/1/exp:	$u_{i,G+1} = x_{i,G} + \lambda \bullet \left(x_{\text{best},G} - x_{r_1,G}\right) + F \bullet \left(x_{r_1,G} - x_{r_2,G}\right)$
4: DE/best/2/exp:	$u_{i,G+1} = x_{\text{best},G} + F \bullet \left(x_{r_1,G} - x_{r_2,G} - x_{r_3,G} - x_{r_4,G}\right)$
5: DE/rand/2/exp:	$u_{i,G+1} = x_{5,G} + F \bullet \left(x_{r_1,G} - x_{r_2,G} - x_{r_3,G} - x_{r_4,G}\right)$
6: DE/best/1/bin:	$u_{i,G+1} = x_{\text{best},G} + F \bullet \left(x_{r_1,G} - x_{r_2,G}\right)$
7: DE/rand/1/bin:	$u_{i,G+1} = x_{r_1,G} + F \bullet \left(x_{r_2,G} - x_{r_3,G}\right)$
8: DE/rand-best/1/bin:	$u_{i,G+1} = x_{i,G} + \lambda \bullet \left(x_{\text{best},G} - x_{r_1,G}\right) + F \bullet \left(x_{r_1,G} - x_{r_2,G}\right)$
9: DE/best/2/bin:	$u_{i,G+1} = x_{\text{best},G} + F \bullet \left(x_{r_1,G} - x_{r_2,G} - x_{r_3,G} - x_{r_4,G}\right)$
10: DE/rand/2/bin:	$u_{i,G+1} = x_{5,G} + F \bullet \left(x_{r_1,G} - x_{r_2,G} - x_{r_3,G} - x_{r_4,G}\right)$.

The convention shown is DE/x/y/z. DE stands for Differential Evolution, x represents a string denoting the solution to be perturbed, y is the number of difference solutions considered for perturbation of x, and z is the type of crossover being used (exp: exponential; bin: binomial).

DE has two main phases of crossover: binomial and exponential. Generally, a child solution $u_{i,G+1}$ is either taken from the parent solution $x_{i,G}$ or from a mutated donor solution $v_{i,G+1}$ as shown : $u_{j,i,G+1} = v_{j,i,G+1} = x_{j,r_3,G} + F \bullet (x_{j,r_1,G} - x_{j,r_2,G})$.

The frequency with which the donor solution $v_{i,G+1}$ is chosen over the parent solution $x_{i,G}$ as the source of the child solution is controlled by both phases of crossover. This is achieved through a user defined constant, crossover *CR* which is held constant throughout the execution of the heuristic.

The *binomial* scheme takes parameters from the donor solution every time that the generated random number is less than the *CR* as given by $\text{rand}_j[0, 1] < CR$, else all parameters come from the parent solution $x_{i,G}$.

The *exponential* scheme takes the child solutions from $x_{i,G}$ until the first time that the random number is greater than *CR*, as given by $\text{rand}_j[0, 1] < CR$, otherwise the parameters comes from the parent solution $x_{i,G}$.

To ensure that each child solution differs from the parent solution, both the exponential and binomial schemes take at least one value from the mutated donor solution $v_{i,G+1}$.

4.2. Self organizing migrating algorithm

SOMA is a stochastic optimization algorithm modeled on the social networking behavior of co-operating individuals [17]. It was chosen because it has been proven that the algorithm has the ability to converge towards the global optimum [17,18].

SOMA works on a population of candidate solutions in loops called *migration loops* or generation loops. The population is initialized randomly which gives it a distribution over the search space at the beginning of the search. In each loop, the population is evaluated and the solution with the highest fitness becomes the leader. Apart from the leader, in one migration loop, all individuals will traverse the input solution space in the direction of the leader.

An individual will travel a certain distance (called the *path length*) towards the leader in n steps of defined length or *step size*. If the path length is chosen to be greater than one, then the individual will overshoot the leader. This path is perturbed randomly.

4.3. Perturbation

Mutation, the random perturbation of individuals, is an important operation for evolutionary strategies (ES). It ensures the diversity amongst the individuals and it also provides the means to restore lost information in a population. Mutation however is applied differently in SOMA compared with other ES strategies. SOMA uses a parameter called PRT to achieve perturbation. This parameter has the same effect for SOMA as mutation has for GA. It is defined in the range [0, 1] and is used to create a perturbation vector (PRT Vector) as follows:

$$\begin{aligned} &\text{if rand}_j < \text{PRT} \quad \text{then PRT Vector}_j = 1 \\ &\text{else } 0, \quad j = 1, \ldots, n. \end{aligned} \tag{14}$$

The novelty of this approach is that the PRT Vector is created before an individual starts its journey over the search space. The PRT Vector defines the final movement of an active individual in search space.

The randomly generated binary perturbation vector controls the allowed dimensions for an individual. If an element of the perturbation vector is set to zero, then the individual is not allowed to change its position in the corresponding dimension.

4.4. Generating new candidate solutions

In standard ES the *Crossover* operator usually creates new individuals based on information from the previous generation. Geometrically speaking, new positions are selected from an N dimensional hyper-plane. In SOMA, which is based on the simulation of cooperative behavior of intelligent beings, sequences of new positions in the N-dimensional hyperplane are generated. They can be thought of as a series of new individuals obtained by the special crossover operation. This crossover operation determines the behavior of SOMA. The movement of an individual is thus given as follows:

$$\vec{r} = \vec{r}_0 + \vec{m}t \ \text{PRTVector} \tag{15}$$

where:

- $\vec{r}$: new candidate solution.
- $\vec{r}_0$: original individual.
- $\vec{m}$: difference between leader and start position of individual.
- $t :\in [0, \text{Path length}]$.
- PRT Vector: control vector for perturbation.

It can be observed from Eq. (15) that the PRT vector causes an individual to move toward the leading individual (the one with the best fitness) in $N - k$ dimensional space. If all N elements of the PRT vector are set to 1, then the search process is carried out in an N dimensional hyperplane (i.e. on a $N + 1$ fitness landscape). If some elements of the PRT vector are set to 0 then the second terms on the right-hand side of Eq. (15) equals 0. This means those parameters of an individual that are related to 0 in the PRT vector are 'frozen', i.e. not changed during the search. The number of frozen parameters, k, is simply the number of dimensions that are not taking part in the actual search process. Therefore, the search process takes place in an $N - k$ dimensional subspace.

5. Chaotic maps

The Lozi map is a two-dimensional piecewise linear map whose dynamics are similar to those of the better known Hennon map [19] and it admits strange attractors.

The advantage of the Lozi map is that one can compute every relevant parameter exactly, due to the linearity of the map, and the successful control can be demonstrated rigorously.

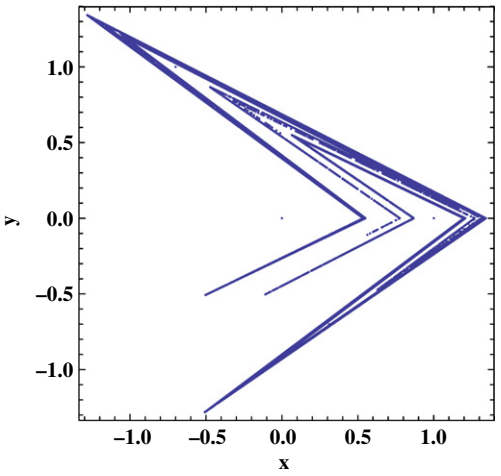

Fig. 2. Lozi map.

Table 1
DE_{chaos} operating parameters.

Parameters	Values
F	0.8
CR	0.9
PopSize	100
Generation	100

The Lozi map equations are given in Eqs. (16) and (17).

$$y_1(t+1) = 1 - a\,|y_1(t)| + y_2(t) \tag{16}$$

$$y_2(t+1) = b y_1(t). \tag{17}$$

The parameters used in this work are $a = 1.7$ and $b = 0.5$ as suggested in [20]. The Lozi map is given in Fig. 2.

6. Results

Experimentation consisted of three systems. The first is the third order system, followed by the fourth order "Ball and Hoop" system [11] and finally the "Electric DC motor" system [9]. Two unique heuristics of DE_{chaos} and $SOMA_{chaos}$ are utilized and the results are compared with the Ziegler–Nichols (ZN) controller. In two instances the results are also compared with published heuristics for that specific problem.

Each experimentation consisted of the maximization of the inverse of the four optimization specifications given in Section 3.1. All four specifications have been used in order to give a balanced result.

The operating parameters for both DE_{chaos} and $SOMA_{chaos}$ were obtained heuristically, over a number of experimentations.

6.1. Third order system

The first problem is a third order system. The plant transfer function is given as:

$$G(s) = \frac{0.1}{s(3s+1)(0.8s+1)}. \tag{18}$$

DE_{chaos} and $SOMA_{chaos}$ operating parameters are given in Tables 1 and 2. These parameters were kept consistent with all the three experimentations.

The objective function values of the four optimization specifications is given in Table 3 for DE_{chaos} and Table 4 for $SOMA_{chaos}$.

From the optimization values of k_p, k_i and k_d, the steady state responses of DE_{chaos} and $SOMA_{chaos}$ are calculated and given in Table 5. The highlighted values present the optimal value for that specification. When compared to the Ziegler Nichols method, DE_{chaos} obtains better performance indices peak overshoot whereas $SOMA_{chaos}$ obtains better results for rise time and settling time.

The system responses for DE_{chaos} for the four different errors is given in Figs. 3–6 and for $SOMA_{chaos}$ in Figs. 7–10.

The combined results of the four different error specifications for DE_{chaos} and $SOMA_{chaos}$ is given in Figs. 11 and 12.

Table 2
$SOMA_{chaos}$ operating parameters.

Parameters	Values
Strategy	All-to-all
PRT	3.0
Stepsize	0.21
Solutions	20
Migrations	20

Table 3
DE_{chaos} objective function values for third order system.

Parameters	Values	k_p	k_i	k_D
IAE	0.0811	5.856	0.0043	11.853
ISE	0.1561	5.204	0.1568	20.804
ITAE	0.0658	4.843	0.00025	7.0234
MSE	31.224	5.2048	0.1568	20.8041

Table 4
$SOMA_{chaos}$ objective function values for third order system.

Parameters	Values	k_p	k_i	k_D
IAE	0.0688	33.477	0.0799	148.74
ISE	0.2605	5.248	1.135	150
ITAE	0.2605	5.248	1.135	150
MSE	0.2605	5.248	1.135	150

Table 5
DE_{chaos} and $SOMA_{chaos}$ steady state responses for third order system.

Specifications	Z–N	DE_{chaos}				$SOMA_{chaos}$			
		IAE	ISE	ITAE	MSE	IAE	ISE	ITAE	MSE
Overshoot	27.7602	10.441	**2.561**	14.36	**2.561**	38.98	29.6	29.6	29.6
Rise time	4.315	4.33	3.335	5.58	3.335	0.99	**0.97**	**0.97**	**0.97**
Settling time	20.315	10.735	26.825	13.085	26.825	**5.66**	41.875	41.875	41.875

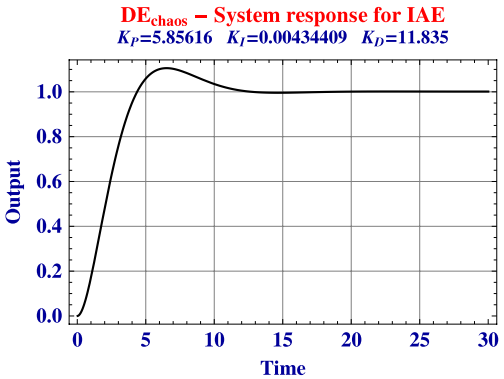

Fig. 3. DE_{chaos} system response for IAE.

6.2. Ball and hoop system

The second problem is a fourth order system comparable with the ball and hoop system [21] and presented in [11]. The Open Loop Transfer Function (OLTS) of this system [11] is given in Eq. (19).

$$G(s) = \frac{1}{s^4 + 6s^3 + 11s^2 + 6s}. \tag{19}$$

The objective functions values of DE_{chaos} and $SOMA_{chaos}$ is given in Tables 6 and 7. The values obtained for both these heuristics are very similar.

The interesting feature is that the values obtained for both the systems is identical. The trajectory of the two systems encompass the same optimal value in the iterative cycle. The steady state responses of the two systems is given in Table 8.

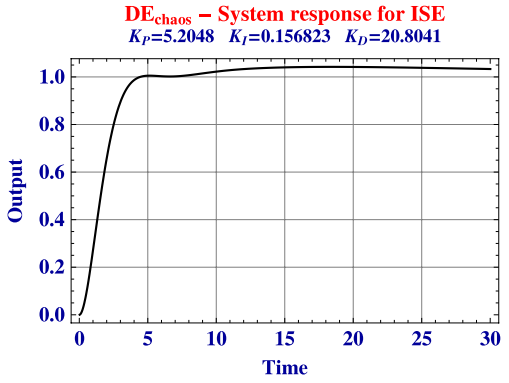

Fig. 4. DE_{chaos} system response for ISE.

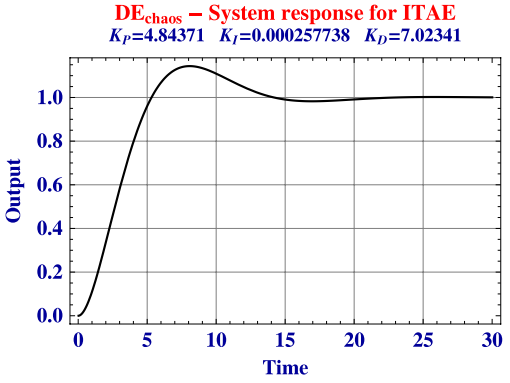

Fig. 5. DE_{chaos} system response for ITAE.

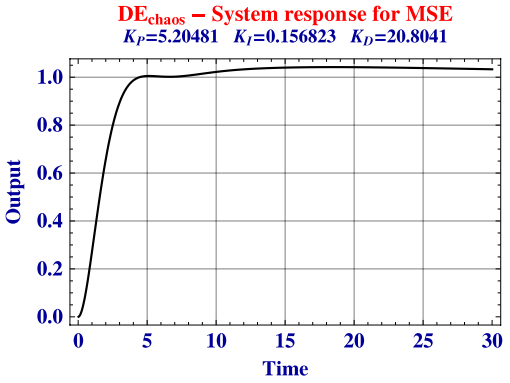

Fig. 6. DE_{chaos} system response for MSE.

Table 6
DE_{chaos} objective function values.

Parameters	Values	k_p	k_i	k_D
IAE	0.0811	5.856	0.0043	11.835
ISE	0.156	5.204	0.1568	20.804
ITAE	0.065	4.8436	0.00025	7.0235
MSE	31.224	5.204	0.1568	20.804

As the system responses of both the chaos map systems is the same, the plots of the different system measures are given in Figs. 13–16. The combined plot is given in Fig. 17.

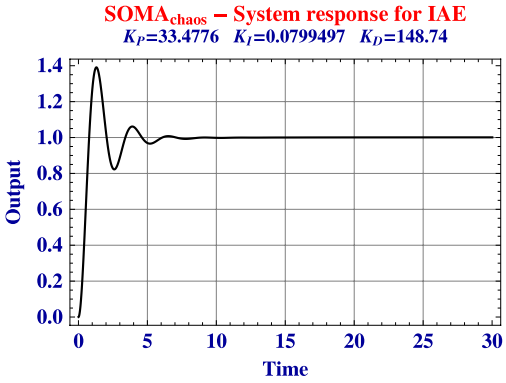

Fig. 7. SOMA$_{chaos}$ system response for IAE.

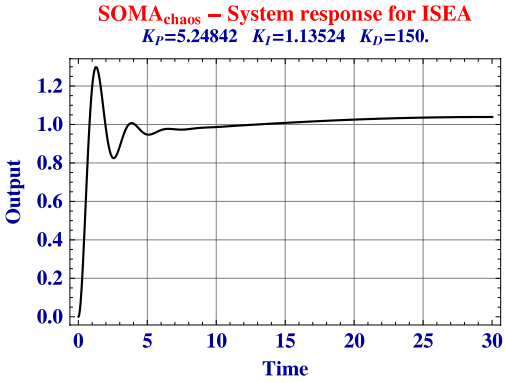

Fig. 8. SOMA$_{chaos}$ system response for ISE.

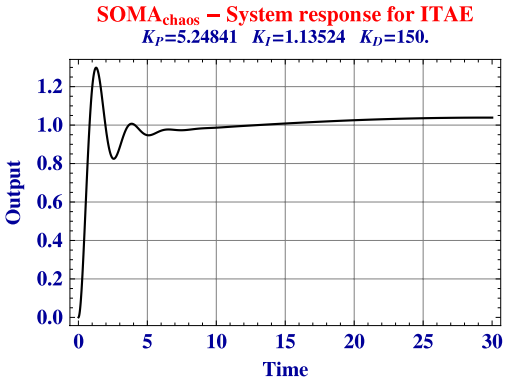

Fig. 9. SOMA$_{chaos}$ system response for ITAE.

Table 7
SOMA$_{chaos}$ objective function values.

Parameters	Values	k_p	k_i	k_D
IAE	0.0811	5.856	0.0043	11.835
ISE	0.156	5.204	0.1568	20.804
ITAE	0.065	4.963	0.0003	7.3089
MSE	31.224	5.204	0.1568	20.804

The results obtained for DE$_{chaos}$ and SOMA$_{chaos}$ are compared with GA of [11] for the same problem. The results are presented in Tables 9 and 10. The results obtained for both DE$_{chaos}$ and SOMA$_{chaos}$ are identical, and apart from the rise time, the chaos variants dominates all other steady state responses over GA.

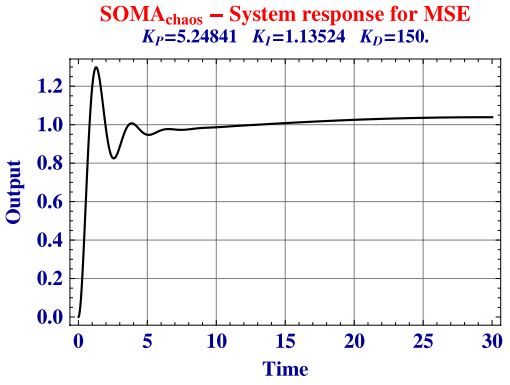

Fig. 10. SOMA$_{chaos}$ system response for MSE.

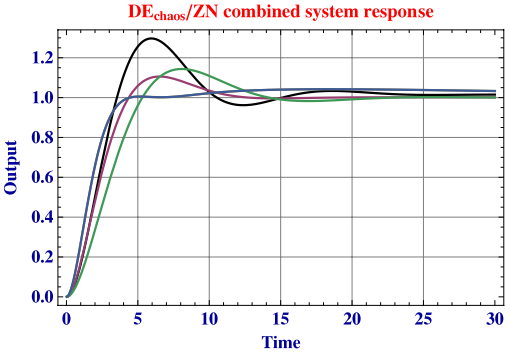

Fig. 11. DE$_{chaos}$ combined system response.

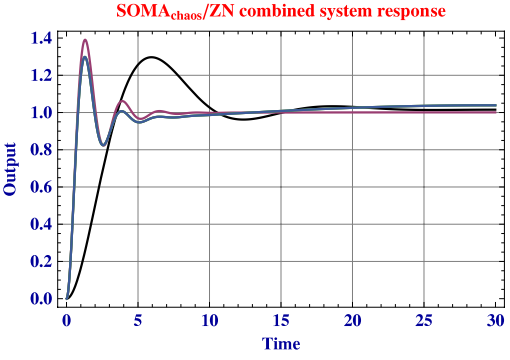

Fig. 12. SOMA$_{chaos}$ combined system response.

Table 8
DE$_{chaos}$ and SOMA$_{chaos}$ steady state responses for fourth order system.

Specifications	Z–N	DE$_{chaos}$				SOMA$_{chaos}$			
		IAE	ISE	ITAE	MSE	IAE	ISE	ITAE	MSE
Overshoot	59.244	14.5	24.52	**6.715**	24.52	14.5	24.52	**6.715**	24.52
Rise time	2.935	1.665	**1.31**	2.155	**1.31**	1.665	**1.31**	2.155	**1.31**
Settling time	15.05	**5.19**	9.22	6.095	9.22	**5.19**	9.22	6.095	9.22

6.3. Electric DC motor system

The third problem is the Electric DC Motor [9]. The transfer function is given in Eq. (20).

$$G(s) = \frac{K}{L_a J s^3 + (R_a J + B L_a) s^2 + (K^2 + R_a B) s} \tag{20}$$

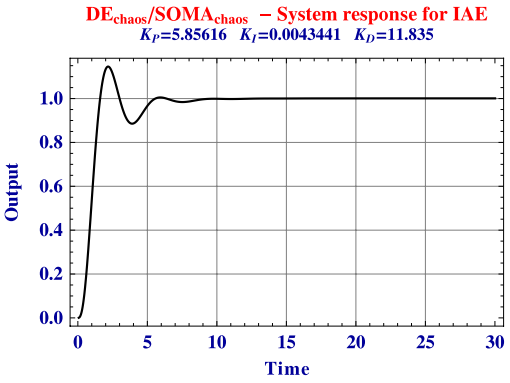

Fig. 13. DE_{chaos} and $SOMA_{chaos}$ system response for IAE.

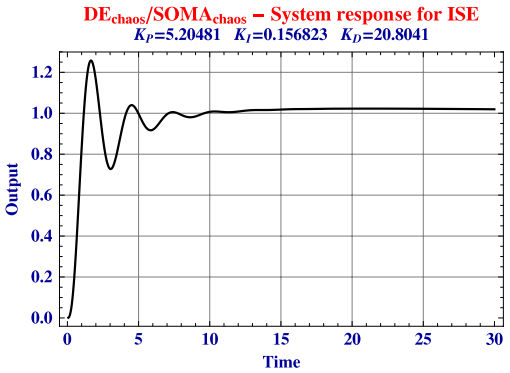

Fig. 14. DE_{chaos} and $SOMA_{chaos}$ system response for ISE.

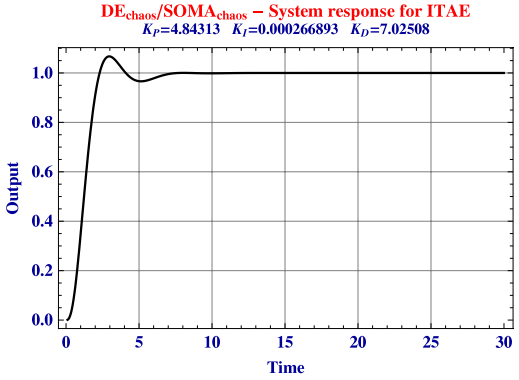

Fig. 15. DE_{chaos} and $SOMA_{chaos}$ system response for ITAE.

Table 9
Comparison of DE_{chaos}, $SOMA_{chaos}$ and GA for ISE and IAE.

Specification	Z–N	IAE			ISE		
		GA	DE_{chaos}	$SOMA_{chaos}$	GA	DE_{chaos}	$SOMA_{chaos}$
Overshoot	59.244	44.97	**14.5**	**14.5**	28.804	**24.52**	**24.52**
Rise time	2.935	**1.2**	1.665	1.665	**1.2**	1.31	1.31
Settling time	15.05	9.3	**5.19**	**5.19**	20.4	**9.22**	**9.22**

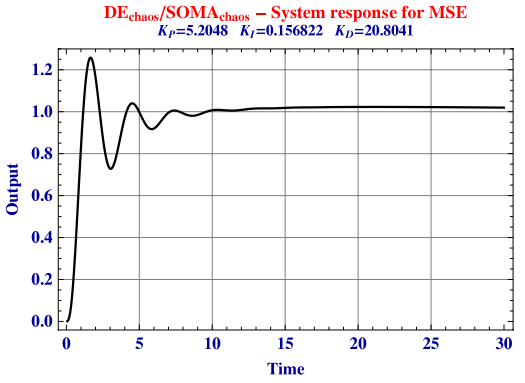

Fig. 16. DE_{chaos} and $SOMA_{chaos}$ system response for MSE.

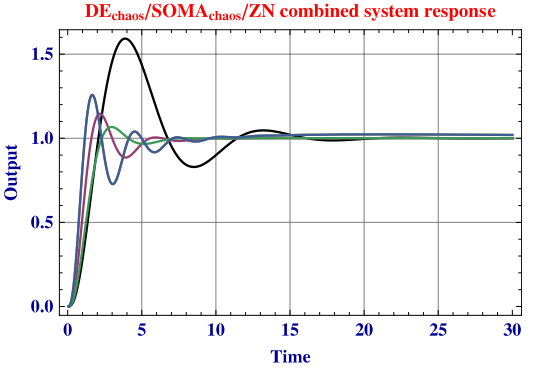

Fig. 17. Combined system response for DE_{chaos}–$SOMA_{chaos}$ compared with Zeigler–Nicholas.

Table 10
Comparison of DE_{chaos}, $SOMA_{chaos}$ and GA for ITAE and MSE.

Specification	Z–N	ITAE			MSE		
		GA	DE_{chaos}	$SOMA_{chaos}$	GA	DE_{chaos}	$SOMA_{chaos}$
Overshoot	59.244	57.19	**6.715**	**6.715**	28.59	**24.52**	**24.52**
Rise time	2.935	**1.3**	2.155	2.155	**1.2**	1.31	1.31
Settling time	15.05	8.2	**6.095**	**6.095**	20.4	**9.22**	**9.22**

where

L_a = armature inductance = 0.025

R_a = armature resistance = 5

K = motor constant = 0.9

J = motor of inertia = 0.042

B = mechanical friction = 0.01625.

The transfer function can now be resolved as in Eq. (21):

$$G(s) = \frac{0.9}{0.0005s^3 + 0.2104s^2 + 0.8913s}. \tag{21}$$

The objective function values of the four specifications are given in Tables 11 and 12.

The steady state responses for DE_{chaos} and $SOMA_{chaos}$ for the DC motor system are given in Table 13.

The system responses for DE_{chaos} for the four different errors is given in Figs. 18–21 and for $SOMA_{chaos}$ in Figs. 22–25.

The combined results of the four different error specifications for DE_{chaos} and $SOMA_{chaos}$ is given in Figs. 26 and 27.

DE_{chaos} and $SOMA_{chaos}$ is compared with the tuning algorithms of *Ziegler–Nichlos* and *Continuous Cycling (CC)* and metaheuristics of *GA*, *Evolutionary Programming (EP)* and *PSO* of [9].

Table 14 gives the steady state responses of all the different heuristics. DE_{chaos} and $SOMA_{chaos}$ are able to produce better results for overshoot and settling time. $SOMA_{chaos}$ obtains the best overshoot value whereas DE_{chaos} has the optimal settling time.

Table 11
DE_{chaos} objective function values.

Parameters	Values	k_p	k_i	k_D
IAE	$3.20351 * 10^{8}$	158.059	$3.333 * 10^{-10}$	36.505
ISE	$2.5533 * 10^{17}$	158.059	$2.57536 * 10^{-7}$	36.505
ITAE	$1.3302 * 10^{9}$	158.059	$8.2886 * 10^{-10}$	36.505
MSE	$5.1066 * 10^{19}$	158.059	$2.5711 * 10^{-7}$	36.505

Table 12
$SOMA_{chaos}$ objective function values.

Parameters	Values	k_p	k_i	k_D
IAE	3.2403	117.608	$5.22 * 10^{-10}$	27.1626
ISE	2.5531	117.607	$1.416 * 10^{-7}$	34.605
ITAE	275 175.71	149.988	1.309	34.6411
MSE	$5.106 * 10^{19}$	117.607	$1.379 * 10^{-7}$	27.162

Table 13
DE_{chaos} and $SOMA_{chaos}$ steady state responses for DC motor system.

Specifications	Z–N	DE_{chaos}				$SOMA_{chaos}$			
		IAE	ISE	ITAE	MSE	IAE	ISE	ITAE	MSE
Overshoot	41.4	11.97	11.97	11.97	11.97	**6.835**	**6.835**	10.475	**6.835**
Rise time	2.56	**0.015**	**0.015**	**0.015**	**0.015**	0.02	0.02	0.02	0.02
Settling time	0.242	**0.035**	**0.035**	**0.035**	**0.035**	0.04	0.04	**0.035**	0.04

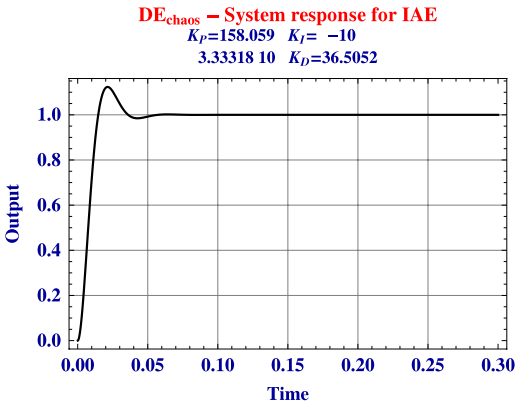

Fig. 18. DE_{chaos} system response for IAE.

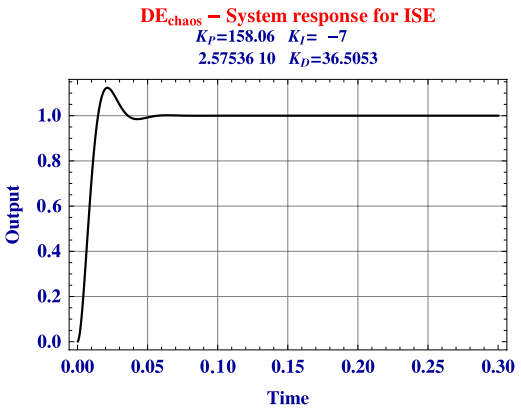

Fig. 19. DE_{chaos} system response for ISE.

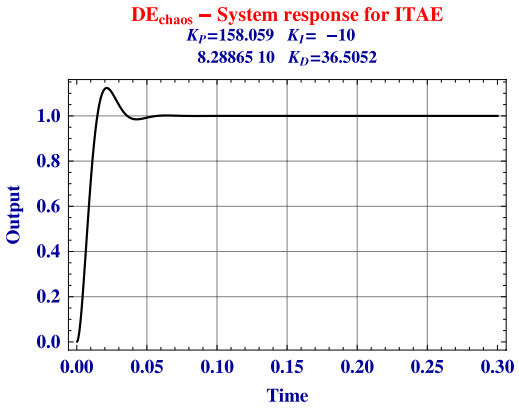

Fig. 20. DE_{chaos} system response for ITAE.

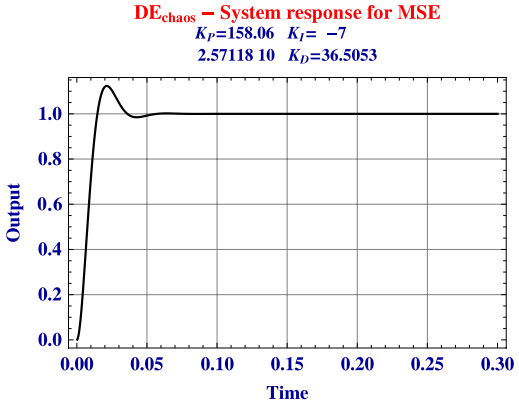

Fig. 21. DE_{chaos} system response for MSE.

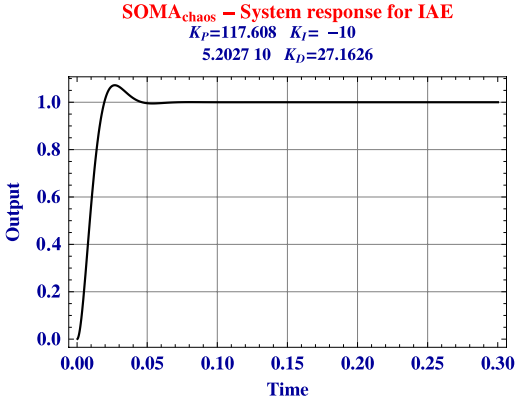

Fig. 22. $SOMA_{chaos}$ system response for IAE.

Table 14
Comparison of steady state responses of different heuristics for the DC motor system.

Specification	Z–N	CC	EP	GA	PSO	DE_{chaos}	$SOMA_{chaos}$
Overshoot	41.4	87.6	8.81	13	12.9	11.97	**6.835**
Settling time	2.56	4.31	0.205	0.324	1.15	**0.015**	0.02
Rise time	0.242	0.0474	**0.014**	0.0317	0.0317	0.035	0.04

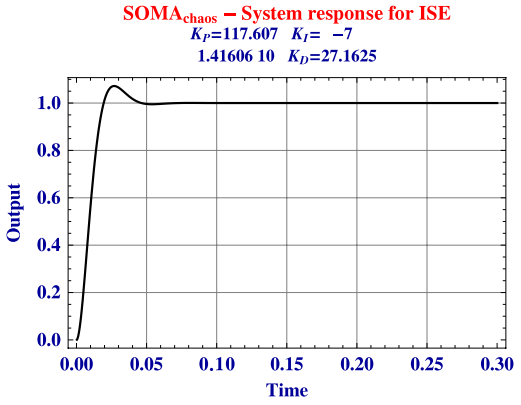

Fig. 23. SOMA$_{chaos}$ system response for ISE.

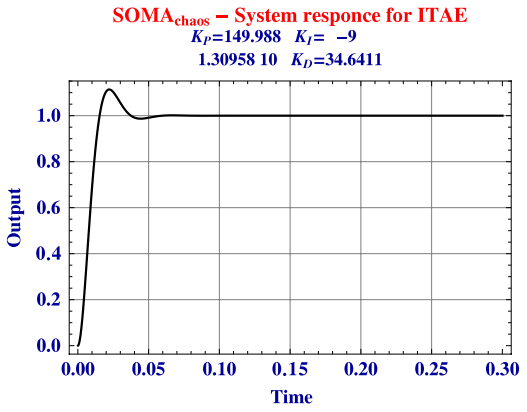

Fig. 24. SOMA$_{chaos}$ system response for ITAE.

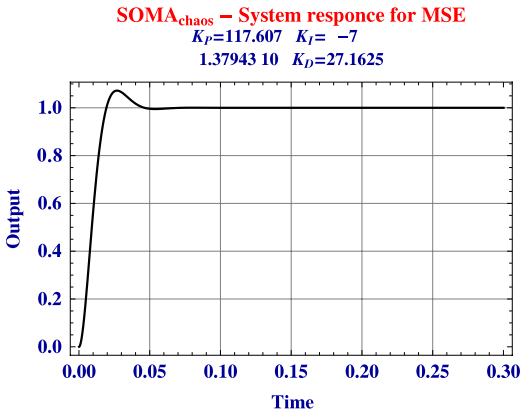

Fig. 25. SOMA$_{chaos}$ system response for MSE.

7. Conclusion

Optimization of control system parameters remains a formidable task due to its complexity. EAs are rapidly becoming a method of choice for some intractable systems. One of the main focal issues is the exploration of suitability of different EAs to this specific task. This paper discusses the application of DE and SOMA to PID controller optimization. The novelty of the approach is the embedding of chaotic systems inside these algorithms to order to provide a superior search mapping structure.

The Lozi map was selected as the chaotic map of choice. This was due to its superior performance in previous research [13]. Three unique PID control problems were selected from literature and evaluated by the chosen heuristics. For all the problems, four unique optimization specifications (IAE, ISE, ITAE, MSE) were utilized. The obtained values were

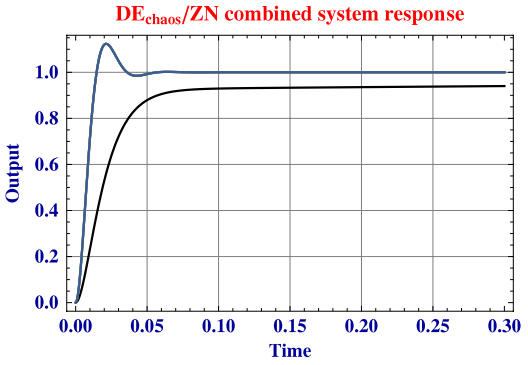

Fig. 26. DE$_{chaos}$ combined system response for DC motor.

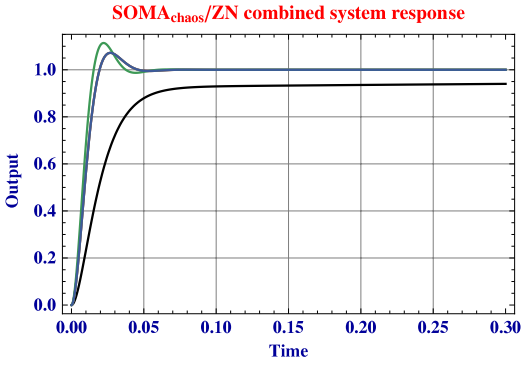

Fig. 27. SOMA$_{chaos}$ combined system response for DC motor.

then used for calculation the three performance indices of rise time, settling time and peak overshoot. The results obtained by chaos driven DE and SOMA, were compared with the values obtained using Ziegler–Nichols method and other published heuristics. For all the attempted problems, DE and SOMA performed better than Ziegler–Nichols method. In the Ball and Hoop system, DE and SOMA performed better than GA, and in the electric DC motor problem, they obtained better overshoot and settling time when compared to other heuristics of GA, PSO, CC and EP. DE and SOMA argument each other favorably, since they both obtained consistently better results.

Acknowledgements

The following two grants are acknowledged for the financial support provided for this research.

1. Grant Agency of the Czech Republic—GACR 102/09/1680.
2. Grant of the Czech Ministry of Education—MSM 7088352102.

References

[1] S. Bennett, The past of PID controllers, in: Proceedings Volume of Digital Control: Past, Present and Future of PID Control IFAC Workshop, Great Britain, 2000, pp. 1–11.
[2] W.-D. Chang, A multi-crossover genetic approach to multivariable PID controllers tuning, Expert Systems with Applications 33 (3) (2007) 620–626.
[3] L.D. Davis, M. Mitchell, Handbook of Genetic Algorithms, Van Nostrand Reinhold, 1991.
[4] Z. Yachen, H. Yueming, On PID controllers based on simulated annealing algorithm, in: Proceedings of the 27th Chinese Control Conference, Yunnan, China, 2008, pp. 225–228.
[5] P. Laarhoven, E. Aarts, Simulated Annealing: Theory and Applications, Kluwer Academic Publishing, Dordrecht, The Netherlands, 1987.
[6] R. Dong, Differential evolution versus particle swarm optimization for PID controller design, in: Fifth International Conference on Natural Computation, Tianjin, China, 2009, pp. 236–240.
[7] M. Chakraborty, D. Maiti, A. Konar, The application of stochastic optimization algorithms to the design of a fractional-order PID controller, in: 2008 IEEE Region 10 Colloquium and the Third ICIIS, Kharagpur, India, 2008.
[8] J.F. Schutte, A.A. Groenwold, A study of global optimization using particle swarms, Journal of Global Optimization 31 (1) (2005) 93–108.
[9] B. Nagraj, S. Subha, B. Rampriya, Tuning algorithms for PID controller using soft computing techniques, International Journal of Computer Science and Network Security 8 (2008) 278–281.
[10] S. Ibrahim, The PID controller design using genetic algorithms, Master's Thesis, University of Southern Queensland, Australia, 2005.
[11] I. Griffin, On-line PID controller tuning using genetic algorithms, Master's Thesis, Dublin City University, Ireland, 2003.
[12] I. Aydin, M. Karakose, E. Akin, Chaotic-based hybrid negative selection algorithm and its applications in fault and anomaly detection, Expert Systems with Applications 37 (7) (2010) 5285–5294.

[13] D. Davendra, I. Zelinka, Controller parameters optimization on a representative set of systems using deterministic-chaotic-mutation evolutionary algorithms, in: I. Zelinka, S. Celikovský, H. Richter, G. Chen (Eds.), Evolutionary Algorithms and Chaotic Systems, Springer-Verlag, Germany, 2010.
[14] K. Astrom, Control System Design, University of California, Santa Barbra, CA, 2002.
[15] Y. Landau, Digital Control Systems, Springer, London, 2006.
[16] K. Price, An introduction to differential evolution, in: New Ideas in Optimization, McGraw Hill, 1999.
[17] I. Zelinka, SOMA—self organizing migrating algorithm, in: G. Onwubolu, B. Babu (Eds.), New Optimization Techniques in Engineering, Springer-Verlag, Germany, 2004.
[18] I. Zelinka, J. Lampinen, L. Nolle, On the theoretical proof of convergence for a class of soma search algorithms, in: Proceedings of the 7th International MENDEL Conference on Soft Computing, Brno, Czech Rep., 2001, pp. 103–110.
[19] M. Hennon, A two-dimensional mapping with a strange attractor, Communications in Mathematical Physics 50 (1979) 69–77.
[20] R. Caponetto, L. Fortuna, S. Fazzino, M. Xibili, Chaotic sequences to improve the performance of evolutionary algorithms, IEEE Transactions on Evolutionary Computation 7 (3) (2003) 289–304.
[21] P. Wellstead, Ball and hoop, 2008, URL: http://www.control-systems-principles.co.uk.