Design of RST Controllers Based on Intelligent Optimization Algorithms

Aman Jacknoon, Mohamed Hassan, Sami El Ferik. *Systems Eng. Dept., KFUPM, KSA*

Abstract*—The Proportional Integral Derivative controller (PID) and the RST controller are the dominant controllers nowadays, since they have simple structure and perform well in most of control loops application. However, unlike PID the RST controller doesn't have direct tuning procedure. In this paper we present an approach for designing and tuning the RST controller by using intelligence optimization algorithms. Three algorithms namely (BSOA, DE and PSO) have been tested in tuning RST controller that controls an electrical DC motor that has been used as a benchmark. Simulation results show PSO tuned RST structure to be more robust and have high performance.*

Keywords—RST Controller, Backtracking Search Optimization Algorithm (BSOA), Differential Evolution Algorithm (DE), Particle Swarm Optimization Algorithm (PSO).

I. INTRODUCTION

In the last decades, the RST controllers have been widely used successfully in many industrial applications, since they demonstrated high robustness and good performance [1], [2].

The RST controller consists of three polynomials named $R(z^{-1})$, $S(z^{-1})$ and $T(z^{-1})$, with two degrees of freedom, i.e., the polynomials R and S are selected to achieve the desired regulation performance, and then the T polynomial is selected to achieve the desired tracking performance. The main problem is how to design these polynomials. In the literature there are two ways to design an RST controller. Either by classical closed loop poles placement technique or by poles-placement with loop shaping of sensitivity function. The poles placement method was proposed by Sylvester as mentioned by Landau in [1], this method can effectively determine the parameters of the RST controller if the plant transfer function is known, or at least the structure of the plant, in the latest case an identification process is required to estimate the plant parameters, however this method will not work effectively with complicated plants. The second method, introduced by Landau in [1], it selects the closed loop poles based on the calibration of the well-known sensitivity functions. However, this method depends on trial and error iteration to select suitable placement of the poles to achieve the desired performance and robustness criteria. Obviously this trial and error method is not the best way to design an RST controller. A systematic approach was introduced by Riadh Madiouni et al [3] to design RST controller using Particle Swarm Optimization algorithm, these intelligence optimization algorithms provide suitable and effective solution to the RST synthesis problem. In this paper an easy and systematic method is proposed using the same Madiouni principle with knowing the plant degree only, three intelligence optimization algorithms namely Backtracking Search Optimization Algorithm (BSOA), Differential Evolution (DE) and Particle Swarm Optimization (PSO). This paper is organized as follows: in section 2 the structure of RST controller is presented and the objective function is formulated, the optimization algorithms are introduced in section 3, the simulation results are given in section 4.

II. PROBLEM FORMULATION

A. RST Controller Structure:

The structure of the RST is presented in Figure 1 below. This structure has two degrees of freedom, as mentioned previously the polynomials R and S are selected to achieve the desired regulation performance, and then the T polynomial is selected to achieve the desired tracking performance.

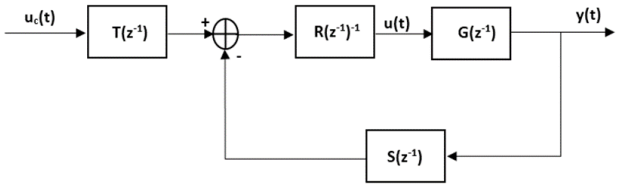

Figure 1: RST Controller Structure.

The RST controller polynomials are given in equation (1):

$$R(z^{-1}) = 1 + r_1 z^{-1} \ldots + r_{n_R} z^{-n_R}$$
$$S(z^{-1}) = s_0 + s_1 z^{-1} \ldots + s_{n_s} z^{-n_s} \quad (1)$$
$$T(z^{-1}) = t_0 + t_1 z^{-1} \ldots + r_{n_T} z^{-n_T}$$

$$G(z^{-1}) = \frac{B(z^{-1})}{A(z^{-1})} = \frac{b_0 + b_1 z^{-1} + \cdots + b_m z^{-m}}{1 + a_1 z^{-1} + \cdots + a_n z^{-n}} \quad (2)$$

Where G is the plant transfer function.

$$y(t) = \frac{BT}{AR + BS} u_c(t)$$
$$u(t) = \frac{AT}{AR + BS} u_c(t) \quad (3)$$
$$AR + BS = A_c \quad (4)$$

The plant is assumed to have a transfer function in the form given in equation (2), the close loop transfer function (3), and the characteristic polynomial is given in (4). As in pole

978-1-5090-1812-3/16/$31.00 ©2016 IEEE

placement method, normally there will be reference model to follow which will determine the desired close loop. The characteristic polynomial is called the Diophantine equation, and it has infinite number of solutions:

$$R = R^0 + QB$$
$$S = S^0 - QA \quad (5)$$

Where Q is an arbitrary polynomial. The solution that *gives* a controller with the lowest degree is called the minimum-degree solution and it is given by selecting:

$$deq\ R = deq\ A - 1 = n - 1$$
$$deq\ A_c = 2\ deq\ A - 1 \quad (6)$$

And to satisfy causality condition and for simplicity we may select that:

$$deq\ R = deq\ S = deq\ T \quad (7)$$

However, other conditions or constrains can be added to achieve specific objective. For example, the R polynomial must be multiplied by (z-1) factor for tracking purpose and steady state error elimination.

There is freedom in the selection of polynomial T, which is selected to achieve the desired tracking performance. Simple and direct selection is by setting T=S, this will reduce the structure of RST controller to a standard lead lag controller:

$$Ru(t) = Tu_c(t) - Sy(t)$$
$$\text{Replace } S = T$$
$$u(t) = \frac{T}{R}\big(u_c(t) - y(t)\big) = \frac{T}{R}\big(e(t)\big) \quad (8)$$

B. Objective Function Formulation:

The main problem in the above explained method (pole-placement) the assumption of having reference model to follow which is the case we want to avoid. Recall that the reference model itself is trying to follow or track the reference input or set-point, thus we can select the set-point as a reference model, and simply by minimizing the error between the system output and set-point we can guarantee that the tracking problem is achieved and the RST controller perfectly performed its job. Integral Square Error (ISE) or Integral Absolute Error (IAE) can be used as an objective function to be minimized [4], [5].

$$IAE(x) = \int_0^\infty |e(x)|dt$$

$$\underset{x=(r_1,\ldots r_n,t_0,\ldots,t_n)}{\text{minmize IAE(x)}}$$

$$\text{Where } \{x \epsilon R^m; x_{min} \le x \le x_{max}\} \quad (9)$$

The optimization algorithms will be used to optimize the controller parameters $(r_1, \ldots, r_n, t_0, \ldots, t_n)$ within the initial search space (x_{min}, x_{max}).

III. Optimization Algorithms

A. Backtracking Search Optimization Algorithm (BSOA):

The Backtracking Search Optimization is a relatively new technique to design an evolutionary algorithm for optimization, it was proposed by Civicioglu [6], and was used in optimization for several applications [7] and [8]. It provided good features such as choosing random particles or solutions from the previous randomly generated population. Civicioglu has elaborated in its applications and compare it with other evolutionary techniques [9]. An improvement has been made to develop adaptive BSOA in order to improve the performance in terms of speed of convergence where mutation and crossover offer more flexibility to the cost function values (i.e. to make it adaptive) [10]. A combination of a BSAO and Burger's chaotic map is used in electrochemical processes [11]. Another application of his technique has been in tuning the PID controller parameters [12]. In [13] BSAO has been used to design a power system stabilizer for multi-machine power systems.

In any optimization procedure it is required to find the optimal value of a parameter θ^* where:

$$\theta^* = \arg\{\min(J(\theta))\} \quad (10)$$

Where $\theta = [\theta_1\ \theta_2\ \ldots.\ \theta_D]'$ is the parameter vector, D is the number parameters to be to be optimized and $J(\theta)$ is the cost function. Following [6] the technique of BSOA includes five steps: initialization, selection-I, mutation, crossover, and selection-II. It can be summarized in the following low chart shown in Figure 2.

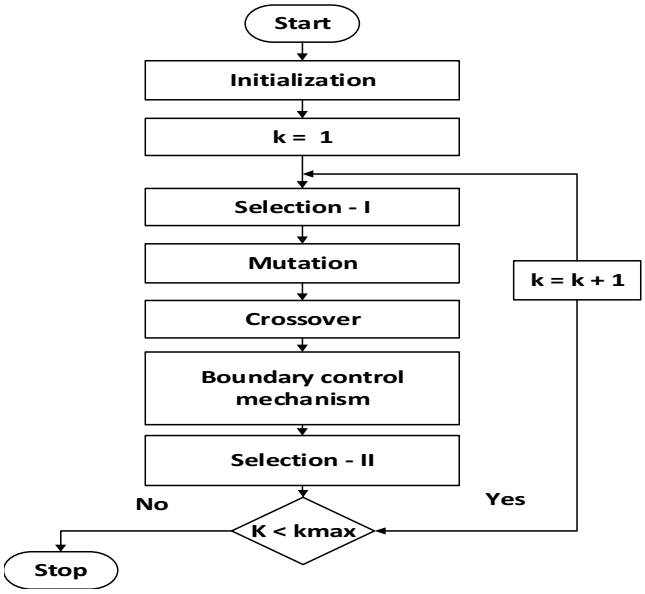

Figure 2: General flowchart of BSOA.

Selection block I evaluates the previous population $oldP_i(k)$ it is computed every iteration randomly.

$$oldP_i(k) = [oldP_{i,1}(k)\ oldP_{i,2}(k)\ \ldots.\ oldP_{i,D}(k)]' \quad (11)$$

The mutation block calculates the initial trial population Mi(k) as:

$$M_i(k) = P_i(k) + F\big(oldP_i(k) - P_i(k)\big);$$
$$i = 1,2,\ldots,N \quad (12)$$

Where $P_i(k)$ denotes the solution at iteration k ($k = 1,2,..,k_{max}$) , kmax is the maximum number of iterations and N is the population size. The crossover block calculates the final trial population Ti(k). The highest fitness trial individual is used to generate next individual population.

The boundary control block recalculates the individuals that are outside the range, and randomly select replacement individuals within the specified range.

The selection-II block uses trial solutions Ti(k) with better fitness values than its equivalent vector Pi(k) so as to update Pi(k).

B. Differential Evolution Algorithm (DE):

The differential evolution, (DE), algorithm is introduced by Rainer Storn and Kenneth Price between 1994 to 1996 [14], [15]. The DE algorithm belongs to an evolutionary algorithms family, it is simple, robust, has fast convergence rate and few control variables. The DE algorithm is suitable for solving non-linear, non-differentiable optimization problems.

The DE algorithm comprises of six steps. The first step, starts with (N_P) initial population generated randomly between two bounds. Each solution comprises of (D) elements which is the dimension of the problem (number of problem parameters needed to be optimized). Additional factors need to be defined, such as generation number or iteration (N_g), mutation factor (F) which control the convergence speed and crossover factor (CR) which plays role in the smoothness of the convergence and also ensures the diversity of the solutions in order not to be trapped in a local minimum during the optimization process.

$$G^i = \left[X_1^i, X_2^i, \dots\dots, X_{N_p}^i\right]$$
$$F \in [0,1], CR \in [0,1] \qquad (13)$$

where i is the generation number, and each solution has (D) parameters $X_n^i = [X_{n1}, X_{n2}, \dots\dots, X_{nD}]$.

In the next two steps, the fitness or objective function for each solution will be calculated, and according to it the best solution among the population will be nominated. In the fourth step the stopping criteria will be checked which may result in terminating or continuing to the fifth step. This step includes mutation and crossover processes. These are the heart of the differential evolution algorithm. Here a variant vector solution (offspring) is generated for each solution in the population by using the following formula:

$$V_i^{(G+1)} = X_i^{(G)} + F\left(X_{best}^{(G)} - X_i^{(G)}\right) + F(X_{r1}^{(G)} - X_{r2}^{(G)}) \qquad (14)$$

Where $Xr_1^{(G)}, Xr_2^{(G)}$ are randomly selected solution vectors from the current generation (different from each other and the corresponding $X_i^{(G)}$and $X_{best}^{(G)}$ is the solution achieving best fitness function. Then a trial solution will be generated by copying the parameters form the parent solution or the offspring solution based on randomly generated probability and the crossover factor, where a random probability number between [0, 1] is generated and then compared to the crossover factor. If the random number is found to be larger than the crossover, then the trial solution will take the parameter from the parent and from the offspring otherwise. In one solution this procedure will be repeated (D) times until the trial solution is formed.

In the last step, there will be N_p trial solutions corresponding to the original population. The fitness function will be calculated for them. The new generation will be formed by comparing the parent solution to the trial solution and takes the one which has the best fitness function as the member (new parent) for the new generation. The whole procedures will be repeated again and again until the stopping criteria is satisfied or the generation (iteration) number is reached.

As long as the number of solutions and iterations gets larger, the possibility to reach the global minimum increases. Since we want to optimize the filter weights, here in our problem the dimension is equivalent to the filter length.

C. Particle Swarm Optimization (PSO):

The PSO algorithm is also an evolutionary algorithm, introduced by Elberhart and Kennedy [16], it was inspired from the swarm behavior of biological populations, the collaboration between the population individuals allow the algorithm to solve complex optimization problems.

The PSO is characterized by its simplicity of concept, ease of implementation, and computationally efficient algorithm [17], [18].

The PSO algorithm is gathering of Np individuals or particles to form the searching swarm, this swarm is distributed randomly on initial search space to find the optimal solution, each particle represents a potential solution has position and velocity given by $x_k^i := \left(x_k^{i,1}, x_k^{i,2}, \dots, x_k^{i,m}\right)^T$ and $v_k^i := \left(v_k^{i,1}, v_k^{i,2}, \dots, v_k^{i,m}\right)^T$ where m is the problem dimensions, k is the generation or iterations number and i is index of solution among its population.at each new generation the individual or particle position is updated according to the following updating rules:

$$x_{k+1}^i = x_k^i + v_{k+1}^i$$
$$v_{k+1}^i = w_{k+1}v_k^i + c_1 r_{1,k}^i\left(p_k^i - x_k^i\right) + c_2 r_{2,k}^i\left(p_k^g - x_k^i\right) \qquad (15)$$

Where: w_{k+1}: inertia factor; c_1, c_2: cognitive and social scaling factors respectively; $r_{1,k}^i, r_{2,k}^i$:are two random numbers in the interval [0,1]; p_k^i is the local best for the i^{th} particle; p_k^g is the global best among the entire swarm.

To improve the searching capacity and convergence rate, the inertia factor is usually evolved linearly with respect to the generation number as shown in [19]:

$$w_{k+1} = w_{max} - \left(\frac{w_{max} - w_{min}}{k_{max}}\right)k \qquad (16)$$

where k_{max} is the maximum iteration number. The general structure of PSO can be summarized in the following points:

i) Define all PSO parameters.
ii) Generate the initial population positions and velocities randomly, and the local best and global best are determined.
iii) Increment the iteration number and apply the updating rules on each particle and evaluate the fitness function then the local best and the global best are updates accordingly.
iv) Check the termination criteria, if it is not satisfied repeat step 3.

IV. Simulation Results

The block diagram shown in Figure 1 has been implemented in Matlab/Simulink environment, the DC motor control problem has been chosen as a benchmark for the proposed method, the DC motor rotates with speed of 3000 rpm when it supplied by 180 volts through an AC-DC power converter. The converter can control the speed by adjusting the armature voltage through pulse width modulation signal (PWM).

The DC Motor transfer function is given in equation 17, discretized version of it is also given, with 10 ms sampling rate [20].

$$G(s) = \frac{k_m}{(1+\tau_m s)(1+\tau_e s)}$$

$$H(z) = \frac{0.0005z+0.0004}{z^2-1.43z+0.4482} \qquad (17)$$

Where the motor parameters nominal values are $k_m = 0.05, \tau_m = 300\ ms, \tau_e = 14\ ms$.

Since the controlled system is second order, the RST controller will be first order, thus it has the following form:

$$R(z^{-1}) = 1 + r_1 z^{-1}$$

$$T(z^{-1}) = S(z^{-1}) = t_0 + t_1 z^{-1} \qquad (18)$$

The objective function will be reduced to:

$$IAE(x) = \int_0^\infty |e(x)|dt$$

$$\underset{x=(r_1,t_0,t_1)}{\text{minmize IAE(x)}}$$

Where $\{x \epsilon R^m; (x_{min} = 0) \leq x \leq (x_{max} = 1)\}$

(19)

The intelligence optimization algorithms (BSOA, DE and PSO) have been developed and applied with the following parameters: (50 population size and 50 number of generation for each of them).

Table 1: Optimization algorithms parameters

DE	
Cross over factor (*CR*)	0.5
Mutation factor (*F*)	0.5
PSO	
Social coefficient c_1	1.2
cognitive coefficient c_2	1.2
Inertia weights range $[w_{min}, w_{max}]$	[0.4,0.9]
N	7
BSOA	
Search amplitude control F	*U(0,3)*

The PSO algorithm succeeded to achieve the best minimum objective value with less time compared to the other two algorithms, Table 2 list these results: (N.B: the integral square errors also calculated although it is not used for optimization).

Table 2: Optimization of the RST controller parameters

	PSO	***BSA***	***DE***
r_1	0.7860	0.039537	0.34405
t_0	0.8865	1	0.98591
t_1	0.3746	0	0.23508
IAE	0.5372	0.5391	0.5404
ISE	0.2831	0.2844	0.3705
RUN Time (sec)	115.5150	116.5034	118.2124

Therefore, the PSO tuned parameters outperform those of the DE and BSOA tuned ones in terms of the quality of the solution and the computation time to reach optimized which is an indicator to a faster response.
In the sense of convergence, generally 30 iterations are sufficient to reach the minimum cost value. Since the number of optimized parameters is few the three algorithms succeed to find their acceptable local minima since the first iteration however they vary from each other in the accuracy of the minimal objective values since DE has proved to be less accurate than PSO and BSOA and PSO have shown best accuracy over the DE and BSOA algorithms. Figures 3 – 5 show the algorithms convergence and Figures 6-8 show their associated RST controller system responses.

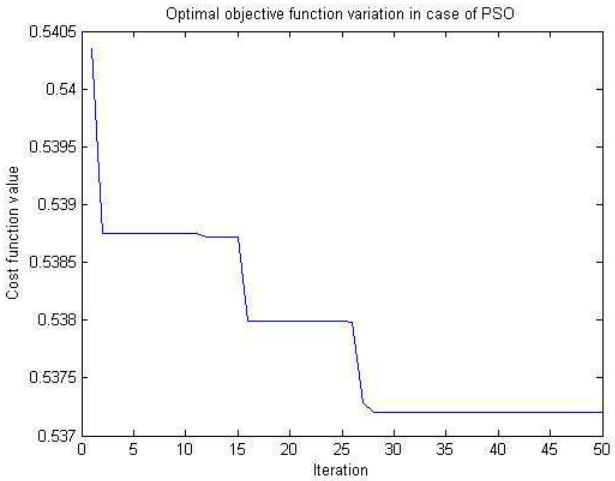

Figure 3: Convergence of the PSO algorithms.

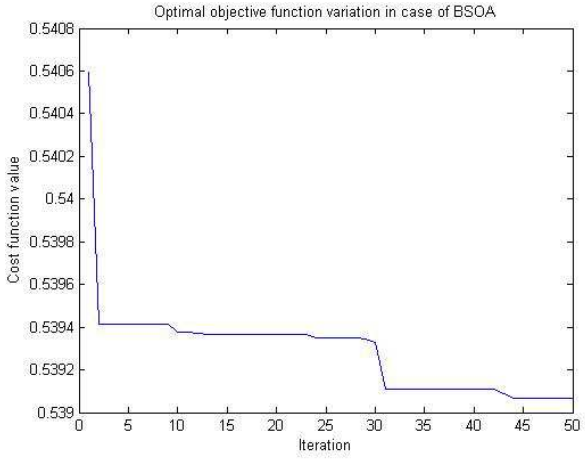

Figure 4: Convergence of the BSOA algorithms.

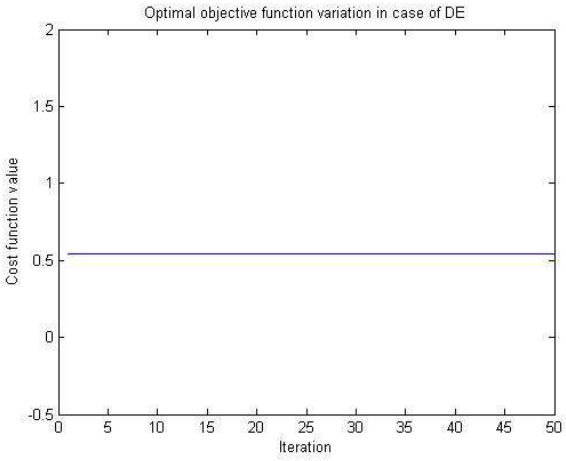

Figure 5: Convergence of the DE algorithms.

As shown in Figures 6, 7 and 8 the time response of the output y(t) when using the PSO tuned parameters, results in less maximum overshoot when compared with the parameters tuned by DE and BSOA. The maximum control effort (represented by the maximum control input amplitude) was shown to be the least in the case of PSO tuned parameters which might be very crucial advantage when we deal with systems having limited actuators. In addition, the error response, e(t), converges to zero faster in the case of PSO tuned parameters.

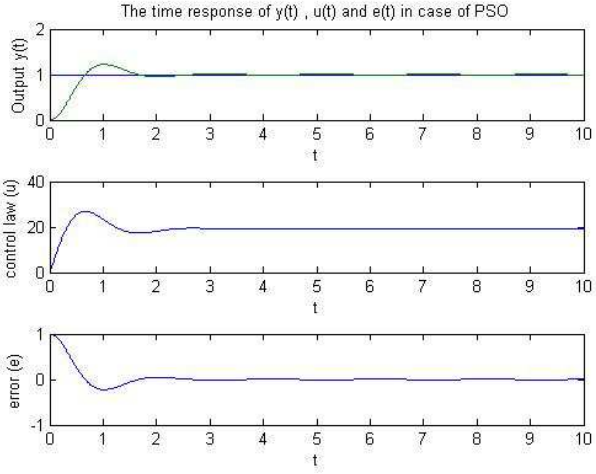

Figure 6: Step response for RST tuned by PSO

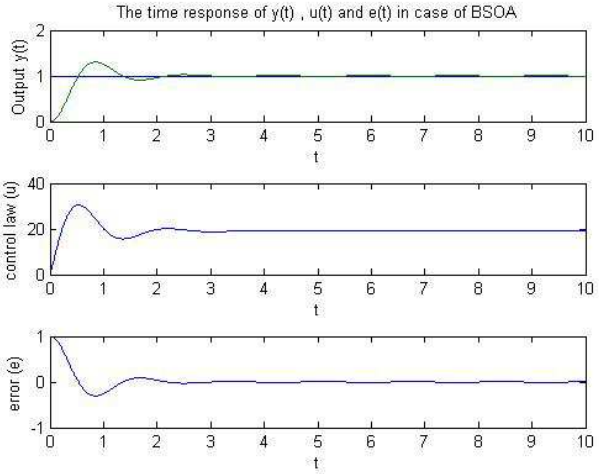

Figure 7: Step response for RST tuned by BSOA

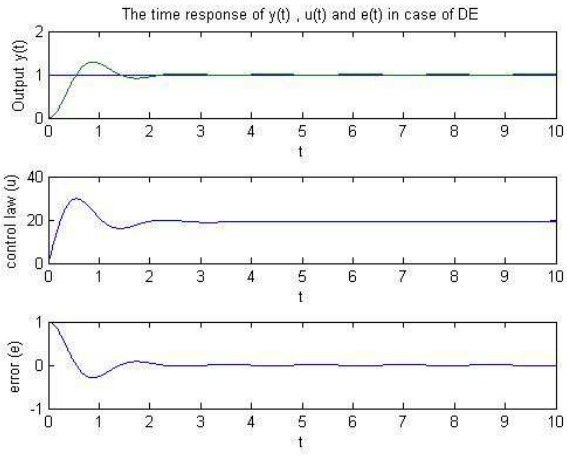

Figure 8: Step response for RST tuned by DE

To show the robustness of the PSO algorithm convergence, the effect of different parameters has been tested, the PSO showed adequate robustness with different parameters variations or combinations as have been shown in Figures 9 and 10 below.

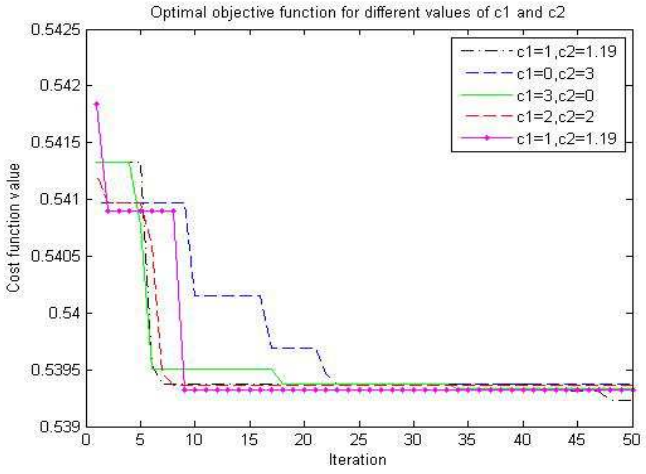

Figure 9: PSO convergence under variations of cognitive and social coefficients

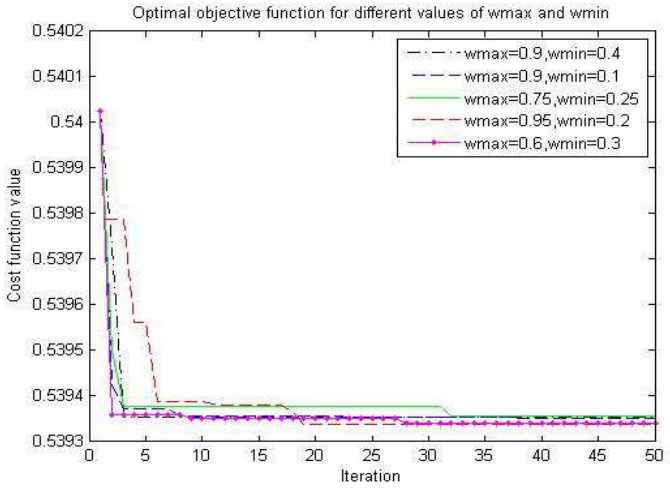

Figure 10: PSO convergence under variations of inertia factor

Figures 9 and 10 reveal that the PSO algorithm is robust to the variation in the cognitive and social coefficients ($c1$ and $c2$ respectively), and in the maximum and minimum value of inertia weight coefficients $Wmax$ and $Wmin$ respectively. Since it exhibits quite similar behavior of cost value decrease along iteration with different values of these parameters.

V. Conclusion:

In this paper a simple tuning technique for reduced RST structure is proposed. DC motor control problem was used as a benchmark, optimization of RST through PSO, DE and BSOA was investigated. PSO shows better performance in terms of achieving the minimum integral absolute error and convergence robustness. The implementation of this technique in real time control is subject of future investigation.

Acknowledgment

The authors would like to thank Mr. Mohanad Ahmed for useful discussion and comments.

References

[1] Landau, Ioan Doré, and Alireza Karimi. "Robust digital control using pole placement with sensitivity function shaping method." International Journal of Robust and Nonlinear Control 8.LA-ARTICLE-2007-036 (1998): 191-210.

[2] Landau, I. D. "The RST digital controller design and applications." Control Engineering Practice 6.2 (1998): 155-165.

[3] Madiouni, Riadh, et al. "Particle swarm optimization-based design of polynomial RST controllers." Systems, Signals & Devices (SSD), 2013 10th International Multi-Conference on. IEEE, 2013.

[4] Bouallègue, Soufiene, et al. "PID-type fuzzy logic controller tuning based on particle swarm optimization." Engineering Applications of Artificial Intelligence 25.3 (2012): 484-493.

[5] Bouallègue, S., J. Haggège, and M. Benrejeb. A new method for tuning PID-type fuzzy controllers using particle swarm optimization. INTECH Open Access Publisher, 2012.

[6] Civicioglu, Pinar. "Backtracking search optimization algorithm for numerical optimization problems." Applied Mathematics and Computation 219.15 (2013): 8121-8144.

[7] Guney, K., A. Durmus, and S. Basbug. "Backtracking search optimization algorithm for synthesis of concentric circular antenna arrays." International Journal of Antennas and Propagation (2014)

[8] Vitayasak, Srisatja, and Pupong Pongcharoen. "The Backtracking Search Algorithm for designing a robust machine layout." Control Engineering and Electronics Engineering 95 (2014): 411.

[9] Civicioglu, Pinar. "Circular antenna array design by using evolutionary search algorithms." Progress in Electromagnetics Research B 54 (2013): 265-284.

[10] Duan, Haibin, and Qinan Luo. "Adaptive Backtracking Search Algorithm for Induction Magnetometer Optimization." Magnetics, IEEE Transactions on 50.12 (2014): 1-6.

[11] Askarzadeh, Alireza, and Leandro dos Santos Coelho. "A backtracking search algorithm combined with Burger's chaotic map for parameter estimation of PEMFC electrochemical model." International Journal of Hydrogen Energy 39.21 (2014): 11165-11174.

[12] Precup, Radu-Emil, et al. "Backtracking Search Optimization Algorithm-based approach to PID controller tuning for torque motor systems." Systems Conference (SysCon), 2015 9th Annual IEEE International. IEEE, 2015.

[13] Shafiullah, Md, M. A. Abido, and L. S. Coelho. "Design of robust PSS in multimachine power systems using backtracking search algorithm." Intelligent System Application to Power Systems (ISAP), 2015 18th International Conference on. IEEE, 2015.

[14] Storn, Rainer, and Kenneth Price. Differential evolution-a simple and efficient adaptive scheme for global optimization over continuous spaces. Vol. 3. Berkeley: ICSI, 1995.

[15] Storn, Rainer, and Kenneth V. Price. "Minimizing the Real Functions of the ICEC'96 Contest by Differential Evolution." International Conference on Evolutionary Computation. 1996.

[16] Kennedy, James, et al. Swarm intelligence. Morgan Kaufmann, 2001.

[17] Poli, Riccardo, James Kennedy, and Tim Blackwell. "Particle swarm optimization." Swarm intelligence 1.1 (2007): 33-57.

[18] Eberhart, Russell C., and Yuhui Shi. "Particle swarm optimization: developments, applications and resources." Evolutionary Computation, 2001. Proceedings of the 2001 Congress on. Vol. 1. IEEE, 2001.

[19] Shi, Yuhui, and Russell C. Eberhart. "Empirical study of particle swarm optimization." Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on. Vol. 3. IEEE, 1999.

[20] Ayadi, Mounir, et al. "A digital flatness-based control system of a DC motor." Studies in Informatics and Control 17.2 (2008): 201.