# Heuristic Model-Free Optimal Controller Design using Gradient Based PSO

Yousef Alipouri
Department of Electrical Engineering, University of Science and Technology, Tehran, Iran
E-mail: yalipouri@iust.ac.ir

Received: March 2014 Revised: December 2014 Accepted: February 2015

**ABSTRACT:**
Designing nonlinear optimal controllers such as Minimum Variance Controller (MVC) has many difficulties. Main difficulties are 1) in order to design controller; the explicit relations between outputs and inputs must be executable. This relation is defined as implicitly in the nonlinear models; 2) learning controller is a high dimensional-multimodal optimization task and search space can be extremely rugged and has many local minima. In this paper, in order to overcome these disadvantages, the model-free optimal controller scheme is utilized. In a model-free controller, as the system model is not available, the gradient of the cost function cannot be executed. Instead, in this paper, a relation between gradient of the controller with gradient of the system model is derived by inverse lemma. The controller structure is selected to be neural network. Then, the gradient based PSO (GPSO) is proposed to be a learning controller. GPSO has both advantages of global searching and convergence properties. The application of the methodology to the empirical CSTR model indicates that this approach gives very credible estimates of the controller. The simulation results indicate that the proposed method can be more accurate than existing methods.

**KEYWORDS:** Nonlinear optimal controller, Neural networks MVC learning, Gradient based PSO, CSTR benchmark system.

## 1. INTRODUCTION

Stochastic search algorithms like PSO perform random walk to explore the search space. A random search allows stochastic optimization algorithms to escape from local minima and explore flat regions but is computationally expensive and leads to slow convergence rates. On the other hand, deterministic algorithms like gradient-based techniques converge faster by using derivative information to identify a good search direction but get stuck in local minima. Also deterministic techniques perform poorly in minimizing functions for which the global minimum is surrounded by flat regions where the gradient is small. Therefore, a hybrid algorithm that uses deterministic techniques with high convergence rates to locate local minima while using stochastic techniques to escape from local minima and explore flat regions is of interest [1].

Also, in the case of a function with a single minimum (unimodal function) the gradient descent algorithm converges faster than stochastic search algorithms because stochastic search algorithms waste computational effort doing a random search. For multimodal functions, the gradient descent algorithm will converge to the local minimum nearest to the starting point. Therefore, for solving the problem of convergence in stochastic algorithms some features of deterministic algorithms can be utilized. The gradient function acts like a map, and shows the direction to the stochastic algorithm. The algorithm starts at an arbitrary point on the cost surface and minimizes along the direction of the gradient to reach the minimum point, which is usually a local minimum. Therefore, if this feature could be utilized, it gives advantage to have the convergence condition in stochastic algorithms.

The PSO has been also combined with deterministic methods in various ways. Izui et al. [2] combined a PSO with gradients, where the members of the swarm were divided into sequential linear programming (SLP) and PSO individuals. In [3], a hybrid algorithm parallel combination of the PSO algorithm with a gradient-based quasi-Newton SQP algorithm is proposed for the optimization of engineering structures. In [4], the PSO algorithm is combined with a conjugate gradient-based local search method for the identification of nonlinear systems. In [5], authors combined the PSO with a quasi-Newton local search method for solving an economic dispatch problem. A local search component has been added to PSO to improve its convergence speed for estimating the parameters of a gene network model [6].

Those methods have a superior capability in terms of a faster convergence, but they cannot be applied to non-differentiable functions. Even worse, in some application, cost is not produced by the mathematical function, for example it may be determined by a program code. Therefore the gradient is not defined for these applications. Besides, it must be noted that stochastic algorithms have a black box view to the cost function. Any information about the gradient of the cost function will not be accepted. Otherwise, the algorithm performance will drop dramatically. This disadvantage can, however, be solved by approximately deriving the gradient.

A method using the gradient by a simultaneous perturbation [7] has previously been proposed as PSO techniques that use the sensitivity of the object function. However, this method has a difficulty in that its effectiveness declines as the number of variables of the object function increases [3]. There is a main concern on accuracy of methods using approximation of the gradient function. Notice that the gradient function is usually approximated by points which may be far from together. Therefore, approximated methods cannot be much effective in promoting the convergence of the algorithm. In this paper, the gradient function is not approximate; instead, in the special application of designing model-free optimal controller, the gradient function is calculated by inverse lemma. In the classic methods of designing optimal controllers, there is need to calculate gradient of a quadratic cost function, and to calculate it there is need to determine the gradient of the system model. However, in model-free control design, there is no information of the system model. In this paper, to overcome this problem, the inverse lemma is utilized. The inverse lemma relates the gradient of the system model to gradient of controller dynamic. Consequently, gradient based PSO can be utilized to optimize the controller.

In this paper, the controller is structured in the form of a neural network. The appearance of neural networks (NNs) helps one considerably in the design of such a needed controller. Theoretically, as long as a sufficient number of neurons are employed, a neural network can approximate a continuous function to an arbitrary accuracy on any compact set [8]. It is thus natural and of practical significance to investigate the proper way of introducing neural networks in nonlinear controller designs for an improved control performance, which is the focus of the paper. The application of the methodology to the empirical CSTR model indicates that this approach gives very credible estimates of the controller. The simulation results indicate that the proposed method can be more accurate than existing methods.

The organization of this paper is as follows: In section II, the proposed algorithm (GPSO) is introduced. Then, in section III, a method to calculate gradient of the cost function is explained.

In section IV, framework of GPSO for optimal model-free controller design will be discussed. At last, section V presents the results of the proposed algorithm to design controller for a CSTR benchmark process.

## 2. GRADIENT BASED PSO (GPSO)

The PSO method is a stochastic algorithm that simulates the movement of flocks of the birds [1]. It can be employed to minimize a general function *J(x)*, where *x* is a vector in a multidimensional space. In this approach a population of individuals (potential solutions of *J(x)*, called particles) update their movements to reach the target point [the global minima of *J(x)*] by continuously receiving information from other members of the flock. In the classical PSO [9], the $n_{th}$ particle velocity and position are updated according to

$$V_{i+1} = wV_i + c_1 r_1 \left(P_L - x_i\right) + c_2 r_2 \left(P_g - x_i\right) \quad (1)$$

and

$$x_{i+1} = x_i + \alpha V_{i+1} \quad (2)$$

Here, $w$ is inertial weight factor, $P_L$ is the local best vector of the n*th* particle, and $P_g$ is the global best vector; $c_1$ and $c_2$ are adjustable social factors; $r_1$ and $r_2$ are random numbers (between 0 and 1); $\alpha$ is the time step.

On the other hand, in the gradient descent method, *J(x)* is minimized by updating the vector $x_i$ according to

$$x_{i+1} = x_i - \alpha \nabla J\left(x_i\right) \quad (3)$$

where, $\nabla J\left(x_i\right)$ is gradient of the cost function which works as the search direction.

In the GPSO algorithm, the PSO algorithm is first used to approximately locate a good local minimum. Then a gradient based local search is done with the best solution found by the PSO algorithm as its starting point. If the best solution found by the PSO algorithm (G) has a larger cost than the final solution found by local search during the previous iteration (L), then L is used as the starting point for the local search. This ensures that the local search is done in the neighborhood of the best solution found by the GPSO in all previous iterations. Thus, the PSO algorithm is used to go near the vicinity of a good local minima and the gradient descent scheme is used to find the local minimum accurately. Next, this accurately computed local minimum is used as the global best solution in the PSO algorithm to identify still better local minima and the cycle is repeated. In this way the GPSO algorithm locates the global minimum by locating progressively better local minima. The main problem in realization of this algorithm is calculating the gradient part (3). Next section proposes a solution to this problem.

## 3. GRADIENT CALCULATION TO USEINMODEL-FREE CONTROLLER DESIGN

Optimality of a controller is measured by a cost function. The most common cost function is defined in the sense of minimum variance as follows [10]:

$$J = E(\eta^2) \qquad \eta = py(t) + qu(t) \tag{4}$$

where $y$ is output, $u$ is control signal, $p$ and $q$ are positive definite (PD) weights and $E(.)$ is expected value operator. Consider a discrete time system given by the nonlinear model as

$$y(t+1) = f(y(t), u(t)) \tag{5}$$

where $y(t)$ is the output, $u(t)$ is the input, and $f(.)$ is the output transition map assumed to be at least continuously differentiable. Assuming the controller is structured as follows:

$$u(t) = g(y(t), \ldots, y(t-d), \theta(t)) \tag{6}$$

where $\Theta(t)$ are the controller unknown parameters, time-dependent in the general case and need to be adaptively optimized. In this case, we propose using an optimization method to determine controller unknown parameters. To implement this idea, suppose an arbitrary structured controller as (6), where $\theta$ (unknown parameters) will be found by GPSO. The gradient descent part of the algorithm is calculated as

$$\theta_{i+1} = \theta_i - \lambda \frac{\partial J}{\partial \theta} \tag{7}$$

where

$$\frac{\partial J}{\partial \theta} = 2E\left( \eta \times \left( p\frac{\partial y(t)}{\partial \theta} + q\frac{\partial u(t)}{\partial \theta} \right) \right) \tag{8}$$

Using Eq. (5) and (6), we have

$$\frac{\partial J}{\partial \theta} = 2\left( (pf + qu(t)) \times \left( p\frac{\partial f}{\partial u} + q \right) \frac{\partial g}{\partial \theta} \right) \tag{9}$$

Then the unknown parameters $\theta$ can be update recursively as follows:

$$\theta_{i+1} = \theta_i - 2\lambda\left( (pf + qu(t)) \times \left( p\frac{\partial f}{\partial u} + q \right) \frac{\partial g}{\partial \theta} \right) \tag{10}$$

To calculate this, there is a fundamental problems: 1) There is a need to calculate $\frac{\partial f(t)}{\partial u(t)}$, where calculating it by considering the system has unknown properties (in this study, the system is considered to be black box) and is nonlinear, and not applicable. To fulfill this problem, we relate the value of above gradient to calculable gradient function $\frac{\partial g(t)}{\partial u(t)}$. As the controller structure is known (ex. neural network), there is not any problem in using gradient of the controller structure. To derive the relation between the two gradient functions we need some assumptions. The following notations are used in the paper: for all $x$ in the domain of arbitrary function $h$:

$$Dh(x) = \frac{\partial h(x,y)}{\partial x};$$

$$D_1 h(x,y) := \frac{\partial h(x,y)}{\partial x};$$

$$D_2 h(x,y) := \frac{\partial h(x,y)}{\partial y}$$

**Assumptions:**

**(1)** Throughout the paper we shall assume that the gradient of $f$ with respect to $y$, $D_1f(y, u)$, is nonsingular everywhere. Note that this is not too restrictive an assumption as it is automatically satisfied for discrete time systems resulting from discretizing continuous time systems that are both forward and backward integrable.

**(2)** System (5) is strongly controllable, i.e. the controllability matrix, $D_2f(x, u)$, has full rank.

**Lemma (1):** For all $x$, $z$ and $u$ we have $u = g(z, x) \Longleftrightarrow z = f(x, u)$.

*Proof:* Assumption (2) guarantees the existence of $g : y \times y \to u$, which uniquely determines $u$ in terms of $x$ and $z$ or $u = g(x, z)$.

**Lemma (2):** the gradients of $g(z, x)$ are given by $D_1g = (D_2f)^{-1}$ and $D_2g = -(D_2f)^{-1}D_1f$.

*Proof:* Consider the mapping

$$\bar{f}(x,u) = (x, f(x,u)) \tag{11}$$

By above assumptions, the gradient of this mapping,

$$D\bar{f}(x,u) = \begin{bmatrix} I_{n\times n} & D_1 f \\ 0 & D_2 f \end{bmatrix} \tag{12}$$

is invertible at any $x$ and $u$. The inverse function theorem [11] implies that $\bar{f}(x,u)$ has a unique *local* inverse $\bar{f}^{-1}(x,z)$ such that $\bar{f}(\bar{f}^{-1}(x,z)) = (x,z)$. Moreover, $D\bar{f}(x,z) = [D\bar{f}(x,u)]^{-1}$. We know from Eq. (12) that

$$[D\bar{f}(x,u)]^{-1} = \begin{bmatrix} I_{n\times n} & D_2 f^{-1} \\ 0 & -(D_2 f)^{-1} D_1 f \end{bmatrix} \tag{13}$$

Therefore, lemma is proved. Notice that $\bar{f}^{-1}(x,z) = (x, g(x,z))$.

To have high performance tracking controller, the best option is that we select the controller to be inverse of system dynamic, or:

$$u(t) = g_\theta(y_d(t+1), y(t)) \Leftrightarrow y(t+1) = f(y(t), u(t)) = y_d(t+1) \tag{14}$$

where $y_d(t+1)$ is desired output value (i.e. reference input). Therefore, we can utilize lemma 2 for system model ($f$) and controller structure ($g_\theta$). Then the unknown parameters $\theta$ of the controller can be updated recursively as follows:

$$\theta_{i+1} = \theta_i - 2\lambda\left(\left(pf + qu\left(t\right)\right)\times\left(p\left(-\left(\frac{\partial g}{\partial y\left(t\right)}\right)^{-1}\frac{\partial g}{\partial y_d\left(t+1\right)}\right)+q\right)\frac{\partial g}{\partial\theta}\right) \tag{15}$$

where $\lambda$ is step length. With supposing the cost function is convex, derivative function is Lipschitz continues, and $\lambda$ is enough small, convergence to a local minimum can be guaranteed. The number of iterations required in the worst case to generate an iterate $\Theta_k$ such that cost value $\|J\| \le \varepsilon$ (for $\varepsilon > 0$ arbitrarily small) is known to be at most $O(\varepsilon^{-2})$ [12]. A complexity analysis for algorithm (38) is available for the case where the objective function is convex, see [13], for instance. In the case, when the algorithm is trapped in local minimum, global searching property of the PSO will be helpful.

## 4. FRAMEWORK OF THE GPSO FOR OPTIMAL MODEL-FREE CONTROLLER DESIGN

In this paper, the controller is structured in form of a two-layer neural network. The neural networks are then trained for a proper number of iteration by GPSO. At each iteration, the controller is applied on loop, and the output variance is calculated for each particle and all the neural networks are ranked based on the values of their fitness indices. Fig. 1 shows the flowchart of the mentioned GPSO procedure when the neural network minimum variance controller NN-MVC designing are adopted.

```
-Creation of the initial population with proper particle dimension
-Generate initial network (controller)
-while not Stop Criterion do
-Let G be the best solution or G=min(f(θ)).
-Produce new particles using Eq. (1) and (2).
- Deterministic derivative based local search with G as starting point using Eq. (15)
for (for all individuals)
- Set up corresponded NN-MVC by weights values defined in individual
- Sample output of loop on control of  NN-MVC
- Evaluate the cost (variance of sampled data)
end for
- Determine global best particle among gradient descent and PSO particles
-Consider new individuals as the new weights of  NN.
-end while
```

**Fig. 1.** Pseudo-code for NN-MVC design by GPSO

A block scheme of the heuristic neural network process model is in Fig. 2.

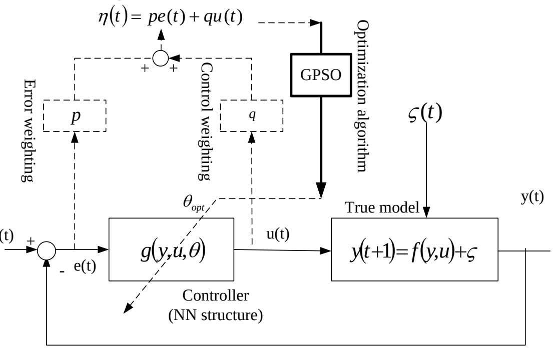


**Fig. 2.** Closed-loop feedback control system structure for the nonlinear plant.

Fig. 3 shows the structure of a two layered feedforward neural network. It shows that the total number of interconnecting weights and thresholds is $((n+m+1) \times p + 2p + 1)$. These data comprise $(n+m+1) \times p$ interconnecting weights between input neurons and hidden neurons, $p$ interconnecting weights between output neurons and hidden neurons, $p$ thresholds of hidden layer, and *one* thresholds of output layer. All of them programmed to be individual with real numbers.

In the encoding strategy, every particle is encoded for a vector. For feedforward neural network (FNN) involved, each particle represents all weights of a FNN's structure. For example, for the FNN with the structure of $(n+m+1)$–$p$–$1$, the corresponding encoding style for each particle with $((n+m+1) \times p + 2p + 1)$ unknown variables can be represented as:

$$particle = \left[w_{11},\dots,w_{(n+m+1)\times p+p},b_1,\dots,b_{p+1}\right] \tag{16}$$

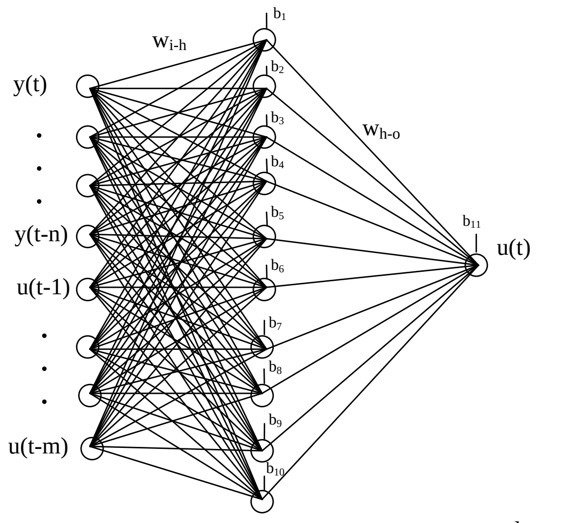


**Fig. 3.** The structure of a two layered feedforward neural network.

## 5. SIMULATION RESULTS

Chemical or biochemical processes are, in general, highly nonlinear, especially when operated over a wide range of operating conditions. The nonlinearity is generally related to reaction kinetics or the nonlinearity of physical properties [14, 15]. Here, the study example is a CSTR with a first-order exothermic reaction provided in [16]. It is a typical chemical engineering process which is intensively studied at the control and system identification areas. The dynamic behavior for this CSTR can be described using the following non dimensional normalized equations:

$$\dot{x}_1 = -x_1 + D_a(1-x_1)e^{\frac{x_2}{1+x_2/\lambda}}$$
$$\dot{x}_2 = -x_2 + BD_a(1-x_1)e^{\frac{x_2}{1+x_2/\lambda}} - \beta(x_2 - x_c) \tag{17}$$

$x_1$ and $x_2$ are the reactor dimensionless concentration and temperature respectively. The case study under consideration is a regulation of outlet reactant concentration $x_1$. Coolant temperatures $x_c$ is the manipulated variable. One set of parameter values $B$ = 1.0, $\beta$ = 0.3, $\lambda$ = 20.0 and $D_a$ = 0.072 which yields an open-loop system with a single stable steady state for all fixed values of the input is selected in [0 23] ([17]). The detailed nomenclature for this exothermic CSTR can be found in [18].

The input range real value range is [0 23]. The output is bounded at range [0 1] for introduced input range [17]. The starting situation is a stable steady situation with the initial states $x_1$ = 0.6219 and $x_2$ = 3.7092 and input $u_t$ = 14. The output $Yt$ with an additive linear disturbance is:

$$Yt = (x_1)t + Dt \tag{18}$$

where $Dt$ is an additive disturbance. The disturbance model is an AR (AutoRegressive) model defined as:

$$Dt = \frac{e_t}{1-0.95q^{-1}} \tag{19}$$

where $e_t$ is a Gaussian white noise with zero mean and variance *0.001*. The realizations of $Yt$ are shown in Fig. 4 for the AR disturbance model. The output variance is *0.01*.

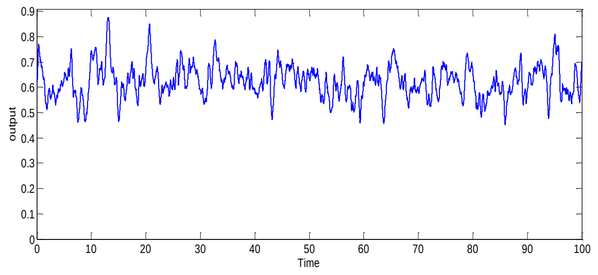


**Fig. 4.** Realizations of open loop control signal and output data.

When the unit proportional feedback controller is been used, the variance of output increases and the output is going to be unstable (Fig. 5).

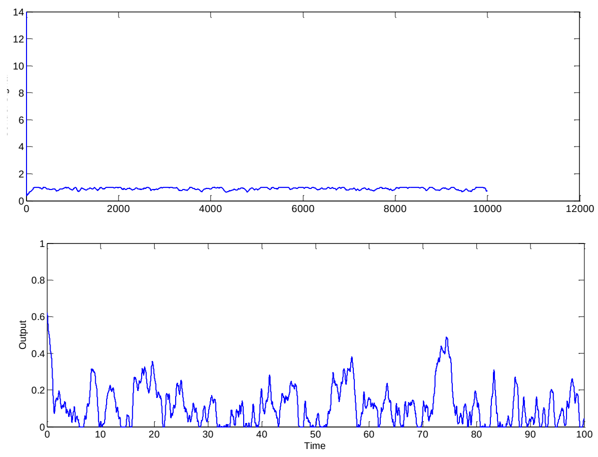


**Fig. 5.** Realizations of proportional closed loop control signal and output data.

NN-MVC has been designed by GPSO method for decreasing output variance of the benchmark system. Population size for GPSO is equaled to *30*; and it is repeated *50* times for reducing the effect of the chance and increasing the reliability in the results. The best results among *50* ones are included in the paper. They are run until the pre-specified generation *100* reached. The initial population is generated randomly with variance 3. The weights values are bounded in range [-10 10]. Fitness of each particle is computed by minimum optimization method (GPSO). Fitness is given by the cost function $J$ for each particle of the population, where $J$ is variance of output as shown in Eq (4).

GPSO is used to evolve the weights of the feedforward neural network with two layered structures. The input layer has *8* nodes (*n=3*, *m=4*); the hidden layer has *10* hidden nodes; output layer has *one* output nodes (Fig. 3). Assuming that the hidden transfer function is sigmoid function, and the output transfer function is a linear activation function. Fig. 6 shows the disturbance, control signal and output data of the NN-MVC designed controller by GPSO. Optimized weights values found by GPSO are included in Appendix A. The minimum variance (minimum cost) found by the algorithm is $9.37\times10^{-5}$.

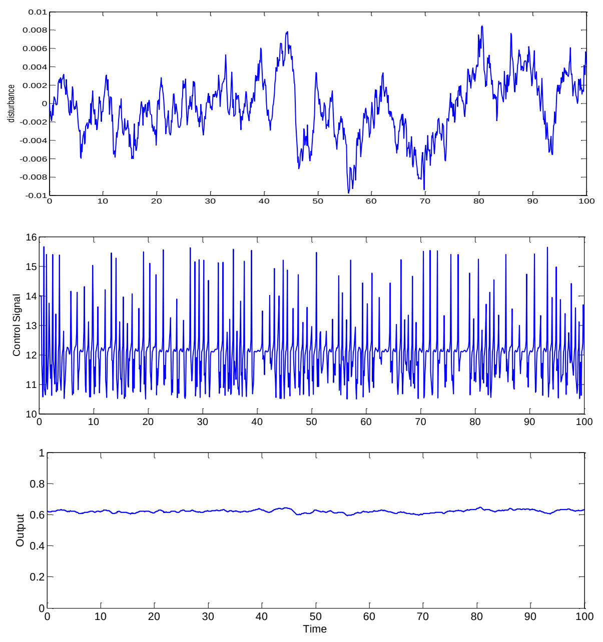


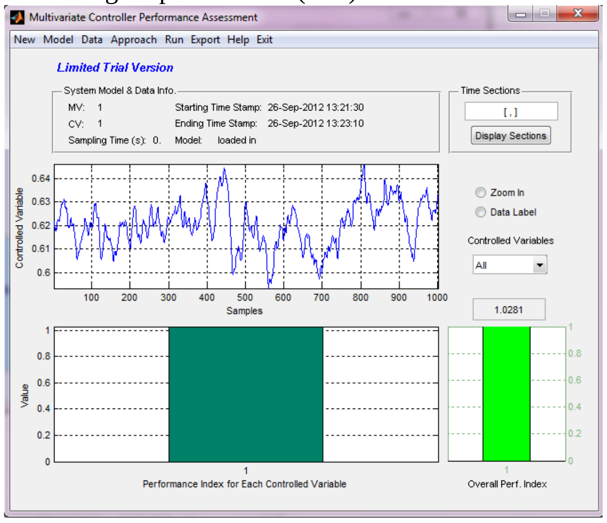
**Fig. 6.** Realizations of disturbance, NN-MVC control signal and output data.

## 5.1. Compare results with performance assessment toolbox [19]

The multivariate controller performance assessment toolbox was developed by the Computer Process Control Group at the University of Alberta to allow performance assessment of linear controller using the Filtering and Correlation (FCOR) Algorithm [19]. Huang et al. [20] developed a filtering and correlation analysis algorithm (FCOR) for MIMO feedback control performance assessment. The control performance index is a single scalar usually scaled to lie within [0, 1], where values close to 0 indicate poor performance, and values close to 1 mean better/tighter control. This indeed holds when perfect control is considered as benchmark. Here, the performance of designed NN-MVC will be assessed by the performance assessment toolbox. The CSTR is linearized around the operating point. Then the linear model and output data of NN-MVC is applied to performance assessment toolbox. Fig. 7 shows the result of assessing the designed controller. It shows that the minimum variance index of the designed controller is *1.0281* which is higher than *1*, so the controller is better than optimal linear MV controller.

## 5.2. Comparing with results reported in [18]:

Above results is admitted by the approach used in [18] for this system in the similar situation, in which linear method reach higher cost ($3.51\times10^{-3}$) than it is reached in this paper ($9.37\times10^{-5}$). In other words, the designed nonlinear controller has been reached to lower minimum variance control than optimal linear method. The results show that the algorithm GPSO significantly improved the model-free controller in decreasing output variance (cost).


Multivariate Controller Performance Assessment

Limited Trial Version

System Model & Data Info.
MV: 1
CV: 1
Sampling Time (s): 0.
Starting Time Stamp: 26-Sep-2012 13:21:30
Ending Time Stamp: 26-Sep-2012 13:23:10
Model: loaded in

Time Sections: [ , ]

Controlled Variables: All

1.0281

New Model Data Approach Run Export Help Exit
Display Sections
Zoom In
Data Label
Controlled Variable
Samples
Value
Performance Index for Each Controlled Variable
Overall Perf. Index

**Fig. 7.** Assessing the designed controller with the performance assessment toolbox which admits the optimality of the controller.

## 6. CONCLUSION

In this paper, the model-free optimal controller scheme is utilized. In a model-free controller, as the system model is not available, the gradient of the cost function cannot be executed. Instead, in this paper, a relation between gradient of the controller with gradient of the system model is derived by inverse lemma. The controller structure is selected to be neural network. Then, the gradient based PSO (GPSO) is proposed to learning the controller. GPSO has both advantages of global searching and convergence properties. The application of the methodology to the empirical CSTR model indicates that this approach gives very credible estimates of the controller. The simulation results indicate that the proposed method can be more accurate than existing methods.

## REFERENCES

[1] M. Noel, **"A new gradient based particle swarm optimization algorithm for accurate computation of global minimum"**, *Applied Soft Computing*, Vol. 12, pp. 353–359, 2012.

[2] K. Izui, S. Nishiwaki, M. Yoshimura, "**Swarm algorithms for single- and multi-objective optimiza**tion problems incorporating sensitivity analysis", *Engineering Optimization*, Vol. 39, No. 8, pp. 981–98, 2007.

[3] V. Plevris, M. Papadrakakis, "**A Hybrid Particle Swarm—Gradient Algorithm for Global Structural Optimization",** *Computer-Aided Civil and Infrastructure Engineering*, Vol. 26, pp. 48–68, 2011.

[4] S. Chen, T. Mei, M. Luo, X. Yang, **"Identification of nonlinear system based on a new hybrid gradient-based PSO algorithm",** in *Proceedings of the International Conference on Information Acquisition*, ICIA. 2007.

[5] L. D. S. Coelho, V. C. Mariani, **"Particle swarm optimization with quasi-Newton local search for solving economic dispatch problem"**, in *Conference Proceedings— IEEE International Conference on Systems, Man and Cybernetics*, 2007.

[6] S. Das, P. Koduru, M. Gui, M. Cochran, A. Wareing, S. M. Welch, B. R. Babin, **"Adding local search to particle swarm optimization",** in *IEEE Congress on Evolutionary Computation*, CEC, 2006.

[7] Y, Maeda, T. Kuratani, **"Simultaneous perturbation particle swarm optimization",** *Proceedings of IEEE Congress on Evolutionary Computation*, Vancouver, BC, Canada,pp. 2687–2691, 2006.

[8] K.. Funahashi, **"On the approximate realization of continuous mappings by neural networks",** *Neural Networks 2*, pp.183-192, 1989.

[9] M.D. Oca, T. Stützle, M. Birattar, M. Dorigo, **"Frankenstein's PSO: a composite particle swarm optimization algorithm",** *IEEE Transactions on Evolutionary Computation*, Vol. 13, 2009.

[10] M. J. Grimble, **"GMV control of nonlinear multivariable systems",** *UKACC Conference Control*, University of Bath, UK, ID-005, 2004.

[11] J. Marsden, *Elementary Classical Analysis*. San Francisco, CA.: Freeman Publishing, 1974.

[12] C. Cartis, N. I. M. Gould, Ph. L. Toint, **"On the complexity of steepest descent, Newton's and regularized Newton's methods for nonconvex unconstrained optimization",** *Siam journal on optimization*, Vol. 20, No. 6, pp. 2833-2852, 2010.

[13] Y. Nesterov, **"Introductory Lectures on Convex Optimization",** *Applied Optimization*, Kluwer Academic Publishers, Dordrecht, The Netherlands, 2004.

[14] D. M. Bates, D. G. Watts, *Nonlinear regression analysis and its applications*, John Willey & Sons, New York, 1988.

[15] R. K. Pearson, B. A. Ogunnaike, **"Nonlinear process identification",** *in* M. Henson and D. Seborg, eds, *`Nonlinear Process Control', Prentice Hall, Upper Saddle River*, N.J., pp. 11-102, 1997.

[16] F. J. Doyle, A. Packard, M. Morari, **"Robust controller design for a nonlinear CSTR",** *Chemical Engineering Science* 44, 1929-1947, 1989.

[17] T. D. Knapp, H. M. Budman, **"Robust control design of non-linear processes using empirical state affine models"**, *Int. J. Control*, Vol. 73, No. 17, pp. 1525-1535, 2000.

[18] W. Yu, **"Variance Analysis For Nonlinear Systems",** PHD thesis, Queen's University Kingston, Ontario, Canada October, 2007.

[19] CPC Control Group, University of Alberta, *University of Alberta Computer Process Control Group*, Multivariate Controller Performance Assessment program, Limited Trial Version, Version 2.1,2010.

[20] B. Huang, S. L. Shah, **"Practical issues in multivariable feedback control performance assessment",** *Proc IFAC ADCHEM, Banff*, Canada, pp. 429–434, 1997.

## Appendix A:

**Table 1**. optimized weights values found by GPSO

| $W_{ij}$ | *j=1* | *j=2* | *j=3* | *j=4* | *j=5* | *j=6* | *j=7* | *j=8* | *j=9* | *j=10* |
|---|---|---|---|---|---|---|---|---|---|---|
| *i=1* | -2.7763 | -6.1925 | -1.9250 | 3.8635 | -9.1758 | 6.4905 | -1.2005 | 1.4717 | -8.5236 | -5.9282 |
| *i=2* | -9.4702 | -3.9335 | 3.8145 | 9.3231 | -9.1604 | -8.2248 | 3.2686 | 2.4480 | -3.8139 | 7.6736 |
| *i=3* | -2.3129 | 3.4643 | -4.2889 | -7.7741 | 9.0812 | 9.7962 | 10.441 | -3.6376 | -2.9007 | 3.3854 |
| *i=4* | -8.8890 | 1.6644 | -1.9433 | -1.3899 | -5.0392 | -7.2601 | 0.4129 | -2.6124 | -2.4283 | -0.7510 |
| *i=5* | -2.8017 | 8.3229 | -2.0793 | 4.4994 | -4.6531 | -5.2277 | -6.6811 | 1.9210 | 1.9698 | 10.059 |
| *i=6* | 8.8527 | 5.9410 | 2.1878 | 8.4467 | 0.6824 | 0.7140 | -5.3913 | 6.1774 | 1.6181 | -10.700 |
| *i=7* | 3.5483 | 9.7259 | 0.0255 | 0.0473 | 3.5124 | -9.5840 | -7.6692 | 7.0161 | 5.6355 | 8.9759 |
| *i=8* | -3.8701 | 6.6945 | 5.9696 | 6.2335 | -1.9470 | 2.2898 | 7.3698 | 1.8640 | -9.8845 | -5.5877 |
| $W_{1j}$ | *j=1* | *j=2* | *j=3* | *j=4* | *j=5* | *j=6* | *j=7* | *j=8* | *j=9* | *j=10* |
| *o=1* | -6.0359 | 4.6606 | -9.6462 | 0.0184 | 3.5209 | 0.5573 | -7.1110 | -8.0571 | -6.6251 | -3.7482 |
| $b_j$ | *j=1* | *j=2* | *j=3* | *j=4* | *j=5* | *j=6* | *j=7* | *j=8* | *j=9* | *j=10* |
| | 8.6359 | 1.5325 | -7.5322 | 3.5676 | -2.8395 | 5.1268 | -4.1412 | 1.5594 | 9.1530 | 4.3419 |
| $b_{11}$ | 9.1845 | | | | | | | | | |

where $w_{ij}$ is the connection weight from the *i*th node of input layer to the *j*th node of hidden layer, $b_j$ is the threshold of the *j*th hidden layer input, $w_{1j}$ is the connection weight from the *j*th hidden node to the output node, $b_{11}$ is the threshold of the output unit.