Research Journal of Applied Sciences, Engineering and Technology 13(11): 871-878, 2016
DOI:10.19026/rjaset.13.3429
ISSN: 2040-7459; e-ISSN: 2040-7467
© 2016 Maxwell Scientific Publication Corp.
Submitted: October 4, 2016 Accepted: November 15, 2016 Published: December 05, 2016

Research Article

Trajectory Tracking Controller of Mobile Robot under Time Variation Parameters based on Neural Networks and Stochastic Fractal Algorithm

Hanan A.R. Akar and Firas R. Mahdi
Electrical Engineering Department, University of Technology, Baghdad, Iraq

Abstract: This study suggests an adaptive Artificial Neural Network (ANN) controller that based on Stochastic Fractal Search algorithm (SFS), the purpose of the Adaptive Neural Controller (ANC) is to track a proposed velocities and path trajectory with the minimum required error, in the presence of mobile robot parameters time variation and dynamical system model uncertainties. The proposed ANC will consist of two sub-neural controllers; the Kinematic Neural feedback Controller (KNC) and the Dynamic Neural feedback Controller (DNC). The external feedback kinematic neural controller is responsible for generating velocity tracking signals that track the mobile robot linear and angular velocities depending on the robot posture error and the desired velocities, while the internal dynamic neural controller is used to enhance the mobile robot against parameters uncertainty, parameters time variation and disturbance noise. The stochastic fractal search algorithm is a Metaheuristic Optimization Algorithm (MOA) that has been used to optimize the Neural Networks (NNs) weight connections to has the behavior of an adaptive nonlinear trajectory tracking controller of a differential drive wheeled mobile robot. The proposed controller has the capability to prepare an appropriate dynamic control left and right torque signals to drive various mobile robot platforms using the same offline optimized weight connections. Metaheuristic optimization algorithms have been used due to theirs unique characteristics especially theirs free of derivative, ability to optimize discretely and continuous nonlinear functions and their ability to get rid of local minimum solution trapping.

Keywords: Artificial neural networks, meta-heuristic algorithms, mobile robot, stochastic fractal search, trajectory tracking controller

INTRODUCTION

Since the first launch of the Artificial Neural Networks (ANNs) and they have been used in many life fields and applications such as; image and signal processing (Lee and Kipke, 2006), systems identification (Kim *et al*., 1994), robotic control (De Sousa Junior and Hemerly, 2000), classification and clustering of data pattern sets. The ANNs have the ability to approximate any nonlinear model by using parallel computation techniques (Hines, 1996). One of the major topics related to the ANNs theory is the learning process of the Neural Networks (NNs). Many algorithms have been used for the learning purpose, one of the most famous and well-known algorithms is the error back propagation algorithm, which has been extensively used for the NNs learning purpose. However, this algorithm appears to be suffering from several problems such as easy being trapped into local minimum solution and its low convergence speed (Gori and Tesi, 1992). Many papers have been made to develop the performance of the back propagation algorithm, while other have just left this concept and migrate to other types of algorithms that called Metaheuristic Optimization Algorithms (MOA) (Siddique and Tokhi, 2001; Rakitianskaia and Engelbrecht, 2009; Bai and Xiong, 2009) especially in the training phase of the NNs.

The neural based controllers of the robotic systems have been gained a great significance in the few recent years. These networks were recommended for their learning ability, intelligent, adaptive behavior and their high performance. An intelligent Mobile Robots (MRs) have become an exciting choice for many scientific types of research and industrial applications. Therefore, a lot of care have been spent to enhance and introduce new controllers, especially in the adaptive field and artificial intelligence.

This study suggests an Adaptive Neural Controller (ANC), that's trained offline using a MOA, called Stochastic Fractal Search algorithm (SFS) (Salimi, 2015) that is motivated by the development of regular phenomenon, inspired from the fractal mathematical idea and from the diffusion feature that seen often in random. The used controller tracks the desired trajectory of a differential drive mobile robot. This

Corresponding Author: Firas R. Mahdi, Electrical Engineering Department, University of Technology, Baghdad, Iraq

controller will be applied to the wheeled mobile robotin the presence of parameters time variation and uncertainties using the same fixed trained weight.

MATERIALS AND METHODS

Adaptive neural controller has been proposed by this study to track a desired designed trajectory, in which the NN has been trained using a recently proposed MOA called SFS algorithm due to its free derivative, faster convergence and ability to escape local minima solution. The proposed controller has the ability to track the trajectory in the presence of noise and parameters time variation also it could be applied to multiple mobile robot platforms with the same fixed optimized weights.

Stochastic Fractal Search (SFS) algorithm: Is a population-based metaheuristic algorithm motivated from the development of regular phenomenon, that uses fractal mathematical concept and the diffusion feature that seen regularly in random. SFS algorithm uses two chief procedures for problem optimization which are: Diffusing and the appraising processes. The diffusing process is in charge for exploitation feature that raises the chance of an agent to catch the global minimum solution and also prevent an agent from being stacked into a locallyoptimal solution. A statically diffusion process is considered, which means that the best agent of the diffusing process is the only agent that will be taken into concern, while other agents are rejected. On the other hand, The appraising process is random approaches that lead to an exploration feature. The Gaussian walks in the diffusion process have been shown by:

$$GS_1 = G(mn_{Bp}, \delta) + (a_1 p_b - a_2 p_i) \tag{1}$$

$$GS_2 = G(mn_p, \delta) \tag{2}$$

$$\delta = \left|\frac{\log(k)}{k}(p_i - p_b)\right| \tag{3}$$

where,

a_1 and a_2	= Two random numbers ∈ $[0,1]$
p_b	= The best solutionagent
p_i	= Anagent in the population
$G(mn, \delta)$	= The Gaussian distribution function with mean (*mn*) and standard deviations (δ).

mn_{Bp} , $mn_p$are equal to p_b and p_i respectively, k is the iteration number, $\frac{log(k)}{k}$ is used to reduce Gaussian jumps size. When all points are randomly initialized, each agent fitness is calculated and the best-obtained agent (p_b) is evaluated. For the purpose of the search space exploitation in the diffusion process, all agents in the population most move around their current location. Two statistical procedures expected to increase the exploration of the search space; the first procedure acts on each different vector index; while the second procedure applied to all agents. During the first statistical procedure, all agents are ranked with respect to the fitness of the agents, then these agents will be given a probability value that follows a simple distribution as shown in Eq. (4):

$$p_{p_i} = \frac{rank(p_i)}{popsiz} \tag{4}$$

Equation (4) shows that the better point has the better probability. Where popsize is the population size of the algorithm. Therefore, it is used to raise the chance of varying the position of agents which have got a bad fitness solution. The l^{th} component of $p_i$agent, is updated according to Eq. (5) if the following condition $p_{pi} < a$ is true, where a is a random number ∈ [0, 1], else it remains unchanged:

$$p_i^{k+1}(l) = p_s{}^k(l) - a(p_u{}^k(l) - p_i{}^k(l)) \tag{5}$$

where, p_s and p_u are two random agents in the population.

In the second statistical procedure, all agents obtained from (5) will be ranked as in (4), if the condition $p_{pi} < a$is true again but for p_i^{k+1}, the recent $p_i^{k+1}$is updated based on Eq. (6) and (7), otherwise no change occurs:

$$p_i^n = p_i^{k+1} - \eta(p_s^{k+1} - p_b)\, for\, \eta \le 0.5 \tag{6}$$

$$p_i^n = p_i^{k+1} + \eta(p_u^{j+1} - p_s^{j+1})\, for\, \eta > 0.5 \tag{7}$$

where, p_s^{k+1} and p_u^{k+1} are two random agents selected from the first statistical procedure and η is a random number formed by Gaussian distribution. p_i^{k+1} will be changed by agent p_i^n if it has better solution fitness (Salimi, 2015). Table 1 will show the general SFS pseudo code.

SFS Optimizing NN weights: As we have seen in the last section, SFS algorithm has madea random population of agents in the search space as the first step toward optimization. The agents, in general, ∈ $R^{popsize*dim}$. The role of SFS algorithm is firstly; To bound and check the candidate solutions in the population, then update the solutions in an iteration process to get the best fitness solution. However, for supervised training NNs the objective function is the Mean Square Error (MSE) function, so that when the weights of the network is fully optimized we will get very minimum or zero MSE.

In order to make the SFS algorithm as a supervised NN learning algorithm, we have considered that the input and hidden layers weights of the NNs are the agent vector in the population, where each agent

Table 1: General SFS pseudo code

```
1:  InitializeSFS algorithm random population and parameters.
2:  While not (best solution is foundor maximum number of
    iteration reached)
3:  For$i$=1 : $popsize$
4:  For $m$ = 1 to maximum diffusion number
5:  Create a new agent based oneq. (1) and (2).
6:  End
7:  End
8:  Use eq. (4) to rank population agents
9:  For $i$=1:$popsize$
10: For each $l$ in $p_i$
11: If $p_{pi} < a$ for $p_i$ then
12: Use (5) to modify $l^{th}$agent
13: Else
14: Keep $p_i$unchanged
15: End if
16: End
17: Use (4) to rank agents of thepopulation.
18: For each $p_i^{k+1}$ in the population
19: If $p_{pi} < a$ for $p_i^{k+1}$then
20: Modify $p_i^{k+1}$ according to (6) and (7)
21: Else
22: Keep $p_i^{k+1}$without modification.
23: End if
24: End
25: End
26: Loop while.
```

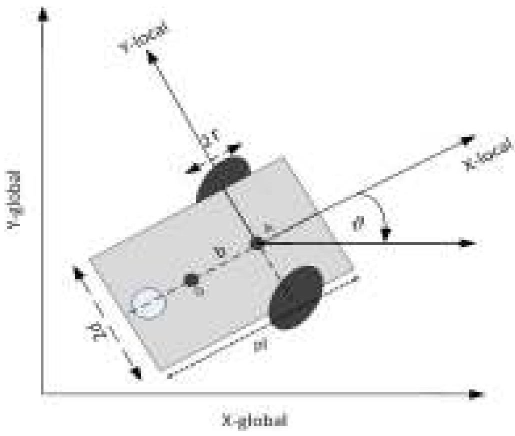

Fig. 1: Differential drive wheeled MR geometrical structure

signifies a candidate weight solution, the purpose of this agent weights vector is to minimize the objective MSE function of the NN. The weights will be bounded between the minimum and maximum search space values. The iteration process will last for a number of epochs (cycle) where each epoch represents a number of iteration process, the original weights will be initialized randomly in the first epoch, while in the next epoch the weights will be the best-obtained agent weights obtained from the last epoch. For stopping iteration criteria, we have put three conditions; if an agent converged to the best solution, or if an over fitting occurred during iteration, or if the maximum number of epochs are reached.

Differential drive wheeled MR modeling: For the kinematic and dynamic MR modeling, we have supposed the model proposed by Fukao *et al.* (2000). Figure 1 shows a general differential drive wheeled MR geometrical structure, in which the point B represents the central point between the driver wheels, b is the straight distance between the center of gravity A and the wheel axis. The MR position will be described by $\rho = [X, Y, \vartheta]^T$, where X and Y are the axis of point A. ϑ is the mobile robot steering rotation angle. The MR kinematic model is given by Eq. (8-10):

$$\dot{\rho} = Q(\rho)\Phi \tag{8}$$

where, $\Phi=[\Phi_r, \Phi_l]^T$, are the right and left wheels angular velocities:

$$Q(\rho)=\begin{bmatrix} \frac{r}{2}\cos(\vartheta) & \frac{r}{2}\cos(\vartheta) \\ \frac{r}{2}sin(\vartheta) & \frac{r}{2}sin(\vartheta) \\ \frac{r}{2}d & -\frac{r}{2}d \end{bmatrix} \tag{9}$$

$$\Phi = \begin{bmatrix} \frac{1}{r} & \frac{b}{r} \\ \frac{1}{r} & -\frac{b}{r} \end{bmatrix} \Lambda \tag{10}$$

where, $\Lambda = [v, w]^T$, are the linear and angular velocities of the wheeled MR.

Substituting Eq. (9) and (10) in (8) we obtain the kinematic system model equations of the differential drive robot in terms of linear and angular velocities as shown in Eq. (11):

$$\dot{\rho} = \begin{bmatrix} cos(\vartheta) & 0 \\ sin(\vartheta) & 0 \\ 0 & 1 \end{bmatrix} \Lambda \tag{11}$$

For non-slipping and pure rolling condition the non-holonomic constraint will be shown in Eq. (12) (Fierro and Lewis, 1998):

$$\dot{X} si\,n(\vartheta) - \dot{Y} co\,s(\vartheta) = 0 \tag{12}$$

Let m be the mass of the robot platform, m_m the mass of the wheel and motor, I the moment of inertia of the robot platform about the perpendicular axis over B, I_m the wheel and motor moment of inertia around the wheel axis and I_i is the wheel and motor moment of inertia around the wheel diameter. The dynamic robot model is represented by Eq. (13-18) as follow:

$$M(\rho)\dot{\Lambda} + N(\rho, \dot{\rho})\Lambda = B(\rho)\tau \tag{13}$$

where, $\tau = [\tau_r, \tau_l]$ stand for the right and left torques applied on the wheels, while M, N, B are represented by:

$$M(\rho) = \begin{bmatrix} \frac{r^2}{4d^2}(m_t b^2 + I_t) + I_m & \frac{r^2}{4d^2}(m_t b^2 - I_t) \\ \frac{r^2}{4d^2}(m_t b^2 - I_t) & \frac{r^2}{4d^2}(m_t b^2 + I_t) + I_m \end{bmatrix} \tag{14}$$

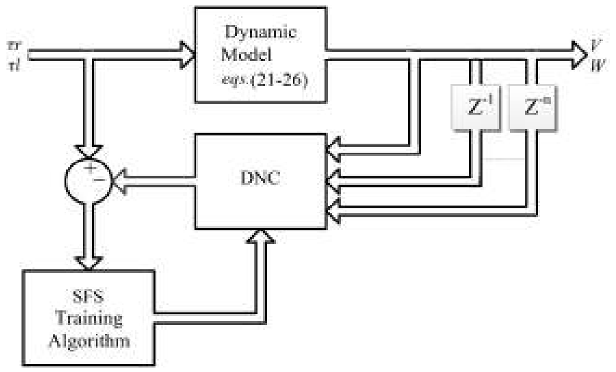

Fig. 2: The DNC training phase

$$N(\dot{\gamma};\gamma) = \begin{bmatrix} 0 & \frac{r^2}{2d}(mbw) \\ -\frac{r^2}{2d}(mbw) & 0 \end{bmatrix} \tag{15}$$

$$B = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{16}$$

where I_t, m_t represent the total moment of inertia and the total mobile robot mass shown in Eq. (17 and 18).

$$I_t = mb^2 + I + 2m_m d^2 + 2I_i \tag{17}$$

$$m_t = m{+}2m_m \tag{18}$$

THE PROPOSED ANC

This study trying to develop a trajectory tracking controller based on NNs that has been optimized using SFS algorithm for the differential drive MR of the Fig. 1. We have assumed that there is uncertainty in the dynamic system model. Furthermore, distance *b,* platform mass *m*, wheel radius *r* and the platform width $p_l$are all varying with time. After we have finished the NN training phase, we assumed the controller will be tough against parameters variation and will track the proposed trajectory at the dynamical and kinematic controller levels.

If we suppose that there is a predefined desired trajectory given by Eq. (19):

$$\dot{\rho_d} = \begin{bmatrix} cos\,(\vartheta_d) & 0 \\ sin\,(\vartheta_d) & 0 \\ 0 & 1 \end{bmatrix} \Lambda_d \tag{19}$$

where, $\rho_d = [X_d, \;\; Y_d, \;\; \vartheta_d]^T$,$\Lambda_d = [v_d, w_d]$. The error between the desired and actual pose in the local robot frame are given by:

$$\begin{bmatrix} E_1 \\ E_2 \\ E_3 \end{bmatrix} = T \begin{bmatrix} e_1 \\ e_2 \\ e_3 \end{bmatrix} = \begin{bmatrix} \cos\,(\vartheta) & \sin\,(\vartheta) & 0 \\ -\sin\,(\vartheta) & \cos\,(\vartheta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_d - X \\ Y_d - Y \\ \vartheta_d - \vartheta \end{bmatrix} \tag{20}$$

The control inputs $V$and$W$ which make E_1, E_2, E_3 converge to zero are given by Eq. (21) (Kolmanovsky and McClamroch, 1995):

$$\begin{bmatrix} V \\ W \end{bmatrix} = \begin{bmatrix} \cos\,(E_3) & 0 \\ k_y E_2 & 1 \end{bmatrix} \begin{bmatrix} v_d \\ w_d \end{bmatrix} + \begin{bmatrix} k_x E_1 \\ k_\vartheta \sin\,(E_3) \end{bmatrix} \tag{21}$$

where, k_x, k_y, $k_\vartheta > 0$, represent positive constants.

The DNC is designed to learn the collected input/output data from the dynamic model system Eq. (13-18) and learns the torque signals which will transfer the MR from velocity at time (*t*) to upcoming (*t+1*) velocity. The DNC offers, after a good training, an adaptive performance with fixed trained weight connections. However, we have to train the DNC for all possible parameters variation combinations of the dynamical model, especially the values *b*, *m*, p_l and *r*. Figure 2 will show the DNC training phase block diagram.

The robot nominal values are proposed as follow; *r* = 0.033 m, *m* = 0.575 kg, p_l = 0.15 m, *b* = 0.04 m. While for the training of the DNC purpose, we have assumed that *m* varied between the values [0.45*,* 1.2] kg, the distance *b* is varied in the interval of [0.03, 0.1] m, p_l varied in between [0.12, 0.2] m, *r* will be varied between [0.03, 0.07] m. The training torques input data sets are proposed randomly uniformly distributed in between [-0.01, 0.01] N.m. On the other hand, the KNC is proposed to learn the behavior of the back stepping feedback controller system Eq. (21) and to raise its robustness against disturbance position data. For the purpose of training the KNC, A random trajectory $\boldsymbol{\rho_r}$ was created and a noisy data having zero mean and 0.01 variation level Gaussian distributed was added to the training data. The training input data of the KNC were the randomly created trajectory reference velocities and the error between the random trajectory and the kinematic model trajectory output plus the noisy Gaussian distributed data. Figure 3 shows the KNC training phase block diagram.

We have generated 10,000 samples data sets, these patterns data sets are divided into two sub-data; 5000

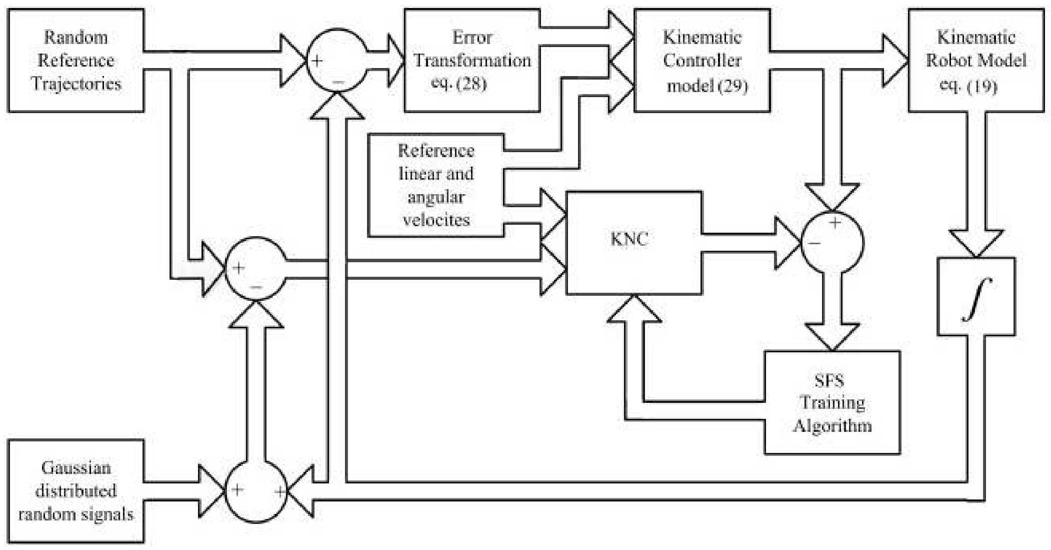
Random Reference Trajectories
Error Transformation eq. (28)
Kinematic Controller model (29)
Kinematic Robot Model eq. (19)
Reference linear and angular velocites
KNC
SFS Training Algorithm
Gaussian distributed random signals

Fig. 3: The KNC training phase

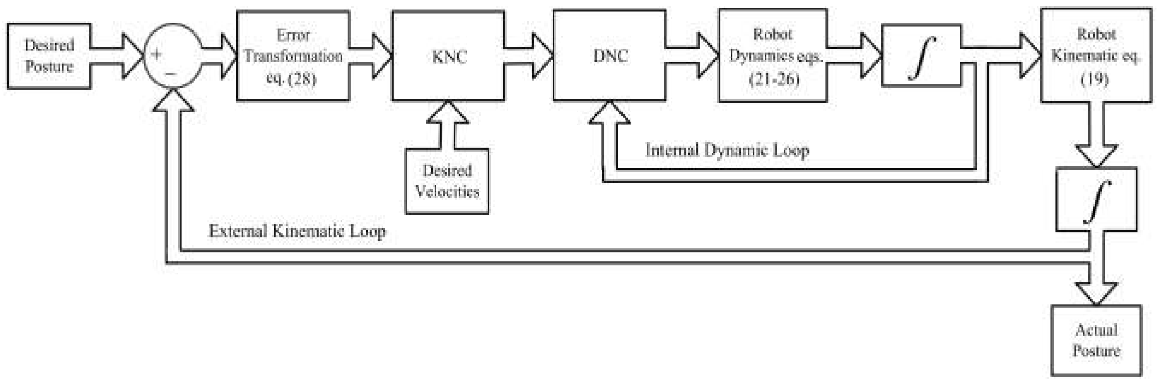

Fig. 4: Overall ANC block diagram

sets have been taken for the training purpose, while the other 5000 sets have been taken for the testing of the trained network. The used NNs architecture for both internal DNC and the external KNC consisted of the single hidden layer with 4 hidden processing neurons and 2 output neurons. The DNC has 4 input neurons for the desired and real velocities, while KNC has 5 input neurons for the random reference velocities and the error posture with disturbance. The synaptic weights connections were randomly initialized between [−1, +1]. Linear identity activation function has been proposed for both input and hidden layers. We have used 10 training epochs (cycle) for both DNC and KNC, each epoch has 1000 maximum iterations (*mit*) and 30 agents population size (popsizes), the SFS algorithm will stop the iterations according to three proposed stopping conditions which are:

- If any agent converged to the predefined global error value, which is zero MSE in our case.
- When the testing MSE exceeds the training MSE by10% of its value.
- When the algorithm completes the whole iteration epochs.

If the algorithm finished an epoch without over fitting the next epoch will continue from the last reached point of the last valid epoch and the network weights will be the last known valid weights.

The overall ANC structure is shown in Fig. 4, where it commonly consists of two feedback control loops; external feedback KNC controller to generate the control velocity signals, that will track the desired trajectories, while the second loop is the internal control loop that will represent the DNC feedback loop that is designed to improve the controller robustness against parameters uncertainty and time variations.

For the purpose of checking the wheeled MR tracking performance, we have assumed that the mobile robot parameters are changing randomly as shown in the Table 2.

The MSE of the MR position and velocities are evaluated by Eq. (20) and (21):

Table 2: Dynamic parameters time variations

Parameters	nominal values	Maximum values	Minimum values
b (m)	0.04 m	0.14 m	0 m
pl (m)	0.15 m	0.5 m	0.1 m
r (m)	0.033 m	0.093 m	0.023 m
m (kg)	0.575kg	3 kg	0.25 kg

Table 3: Velocities trajectories

t	v_d	w_d
$0 \leq t \leq 157.2$	$0.05(1-\cos\left(\frac{3t}{10}\right))$	$0.05(1-\cos\left(\frac{2t}{10}\right))$
$157.3 \leq t \leq 300$	$0.05(1-\cos(\frac{3t}{10}))$	$-0.05(1-\cos(\frac{2t}{10}))$

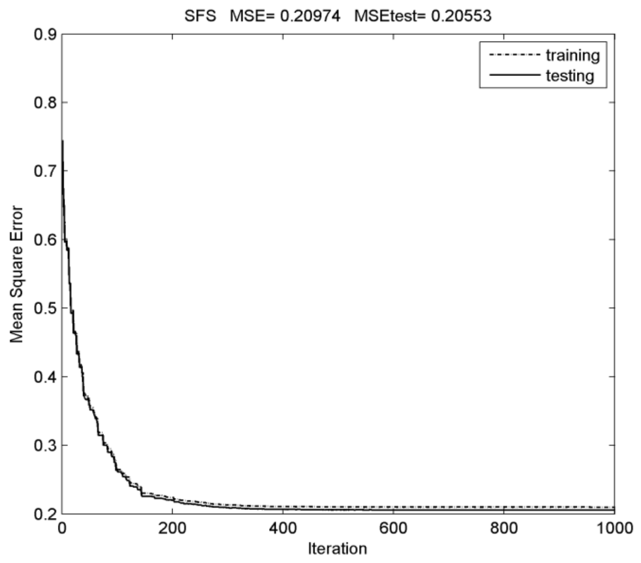

Fig. 5: First epoch SFS training algorithm, KNC

$$Errp = \frac{1}{3}\sqrt{E_1^2 + E_2^2 + E_3^2} \tag{20}$$

$$Errv = \frac{1}{2}\sqrt{(v_d - v)^2 + (w_d - w)^2} \tag{21}$$

The desired velocities generated for the path tracking are shown in the Table 3.

The desired posture will be found integrating Eq. (19).

RESULTS AND DISCUSSION

The SFS algorithm first training epoch for KNC and DNC are shown in Fig. 5 and 6.

After 10 training epochs without overfitting of the KNC we have recorded maximum MSE of 0.00019 for the linear velocity while 0.40611 for the angular velocity, the average angular and linear velocities MSE was 0.2031 which is relatively high MSE, due to the large set of training input sets, the average testing MSE was 0.203. For the second DNC network, we have recorded 2.31e-10 MSE for the left torque signal and 2.09e-10 for the right torque signal, the average MSE of both torques was equal 2.2e-10 with testing average MSE of 2.174e-10. This very low value actually is due to the tiny torque signals that have amaximum value of 0.0001 N.m. The mobile robot position MSE (*Errp*), velocities MSE (*Errv*) are shown in Fig. 7 and 8.

The velocity and the position MSE for the MR with nominal parameters without time-varying and the time varying parameters are less than 4-03. Figure 9 shows the real and desired x-y trajectories tracking.

CONCLUSION

In this study we have proposed an inverse ANC trajectory tracking wheeled MR that consists of two subs NNs; KNC and DNC using two internal and external feedback loops; the internal DNC feedback loop will make the robot more robust against parameters uncertainty and parameter time variations, the second external KNC was responsible for tracking the desired angular and linear velocities of the WMR and hence the posture X, Y and ϑ. We have trained both of NNs using a recently proposed meta-heuristic population based SFS algorithm, applying 10 iteration epochs each with 1000 maximum iterations for both of networks. We have used as minimum as possible hidden processing neurons, getting the higher

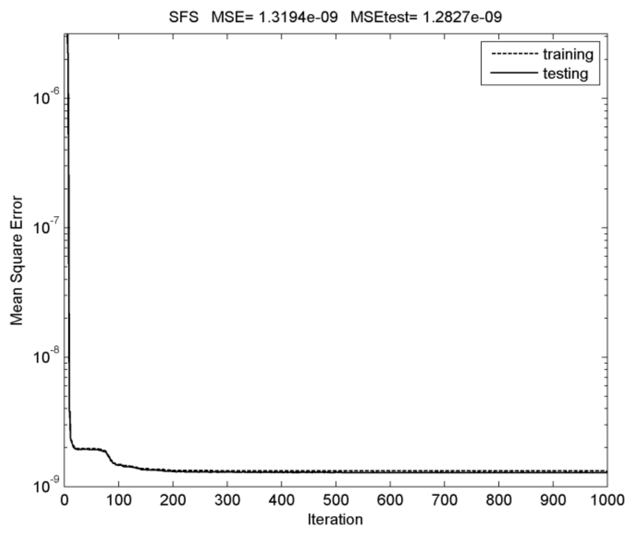

Fig. 6: First epoch SFS training algorithm, DNC

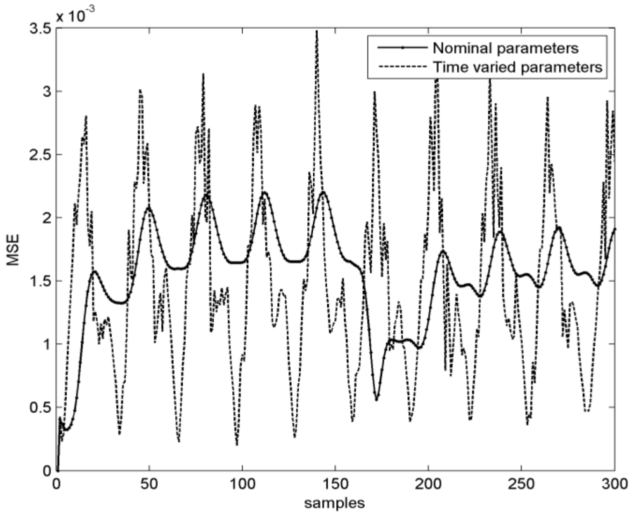

Fig. 7: Mobile robot *Errp* MSE

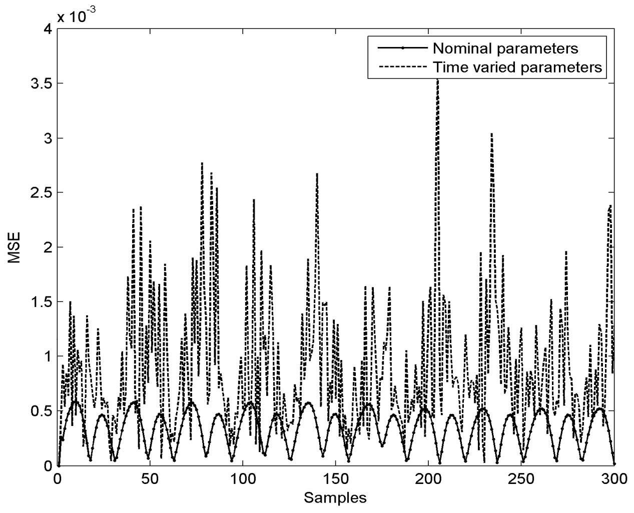

Fig. 8: Mobile robot *Errv* MSE

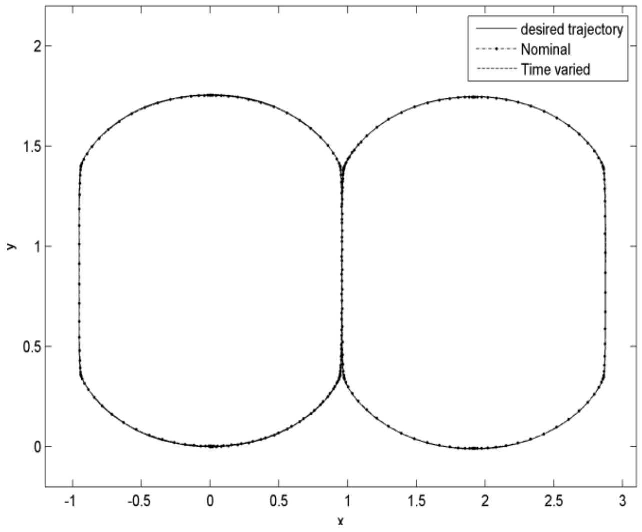

Fig. 9: X-Y trajectory tracking

possible performance. Figure 7 and 8 show the position and velocity MSE, from these figures we notice that the controller succeeded in tracking the MR under fixed nominal parameters values and with dynamic parameters time variation. The ANC performance generally doesn't affected by the parameters variation even when some parameters have been increased by more than 400%. Therefore this controller has shown a great performance against parameters variation with only 8 hidden neurons for both KNC and DNC and without any over fitting during data training process using the SFS algorithm due to the used techniques to observe the training and testing data at the same time. Since the training was offline and have been carried only once (10 epochs), we haven't need more training therefore, we haven't take the SFS processing time into consideration.

REFERENCES

Bai, K. and J. Xiong, 2009. A method of improved BP neural algorithm based on simulated annealing algorithm. Proceeding of the 3rd International Conference on Genetic and Evolutionary Computing (WGEC '09). Guilin, pp: 765-768.

De Sousa Junior, C. and E.M. Hemerly, 2000. Neural network-based controllers for mobile robot. Proceeding of the 6th Brazilian Symposium on Neural Networks. Rio de Janeiro, pp: 50-55.

Fierro, R. and F.L. Lewis, 1998. Control of a nonholonomic mobile robot using neural networks. IEEE T. Neural Networ., 9(4): 589-600.

Fukao, T., H. Nakagawa and N. Adachi, 2000. Adaptive tracking control of a nonholonomic mobile robot. IEEE T. Robotic. Autom., 16(5): 609-615.

Gori, M. and A. Tesi, 1992. On the problem of local minima in backpropagation. IEEE T. Pattern Anal., 14(1): 76-86.

Hines, J.W., 1996. A logarithmic neural network architecture for unbounded non-linear function approximation. Proceeding of the IEEE International Conference on Neural Networks. Washington, DC, 2: 1245-1250.

Kim, S.W., S.G. Hong, T.D. Ohm and J.J. Lee, 1994. Neural network identification and control of unstable systems using supervisory control while learning. Proceeding of the IEEE International Conference on Neural Networks and IEEE World Congress on Computational Intelligence. Orlando, FL, 4: 2500-2505.

Kolmanovsky, I. and N.H. McClamroch, 1995. Developments in nonholonomic control problems. IEEE Contr. Syst., 15(6): 20-36.

Lee, J. and D. Kipke, 2006. Neural signal processing using discrete wavelet transform for neural interfaces. Proceeding of the International Conference on Microtechnologies in Medicine and Biology. Okinawa, pp: 169-172.

Rakitianskaia, A. and A.P. Engelbrecht, 2009. Training neural networks with PSO in dynamic environments. Proceeding of the IEEE Congress on Evolutionary Computation (CEC'09). Trondheim, pp: 667-673.

Salimi, H., 2015. Stochastic fractal search: A powerful metaheuristic algorithm. Knowl-Based Syst., 75: 1-18.

Siddique, N. and M.O. Tokhi, 2001. Training neural networks: Backpropagation vs genetic algorithms. Proceeding of the International Joint Conference on Neural Networks (IJCNN '01). Washington, DC, 4: 2673-2678.